

Bluetooth Low Energy-Based Indoor Positioning System for Mobile Devices

Philipp Hager



MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Embedded Systems Design

in Hagenberg

im September 2023

Advisor:

FH-Prof. Dr.-Ing. habil. Hans-Georg Brachtendorf
Dr.-Ing. Christian Gentner

© Copyright 2023 Philipp Hager

This work is published under the conditions of the Creative Commons License *Attribution-NonCommercial-NoDerivatives 4.0 International* (CC BY-NC-ND 4.0)—see <https://creativecommons.org/licenses/by-nc-nd/4.0/>.

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere. This printed copy is identical to the submitted electronic version.

Hagenberg, September 14, 2023

Philipp Hager

Contents

Declaration	iv
Acknowledgements	vii
Abstract	viii
Kurzfassung	ix
1 Introduction	1
1.1 Aim of this Thesis	1
1.2 Structure of this Thesis	2
2 Fundamentals of Radio-Based Positioning	3
2.1 Radio-Based Positioning	3
2.1.1 Trilateration	3
2.1.2 Bluetooth Low Energy (BLE)	4
2.1.3 Path Loss and Multipath Propagation	7
2.2 State Estimation of Non-Linear Systems	8
2.2.1 Introduction to Recursive Bayesian Filtering	9
2.2.2 Particle Filter Basics	11
2.2.3 Resampling	13
2.2.4 Sampling Importance Resampling Particle Filter	15
3 Concept of a BLE-Based Indoor Positioning System	17
3.1 System Architecture of an Indoor Positioning System	17
3.2 Hardware and Software Platforms	19
3.3 Determination of the Path Loss Model	19
3.4 Server-Based Distance Estimation	22
3.5 Transition Models	24
3.5.1 Gaussian-Transition-Model	24
3.5.2 Inertial Measurement Unit (IMU)-Transition-Model	25
3.6 Determination of the Orientation of a Mobile Device	27
3.7 Particle Filter-Based Position Estimation	28
3.8 Further Measures to Enhance the Position Estimation	29

4	Implementation of the Server-Based Indoor Positioning System	31
4.1	Matlab Version	31
4.2	nRF52840 Dongle Firmware	31
4.3	Android Application	33
4.3.1	BLE Beacon	33
4.3.2	Integration of the Inertial Data Fusion	34
4.4	Embedded Implementation	35
4.4.1	Server-Based Architecture	36
4.4.2	Sniffer Implementation	36
4.4.3	Server Implementation	38
4.5	Offline and Live Positioning Mode	40
5	Evaluation of the Server-Based Indoor Positioning System	42
5.1	Testbed and Experimental Setup for Data Acquisition	42
5.2	Preparation of the Recorded BLE and Ground Truth Data	43
5.2.1	Individual Path Loss Model Calibration	43
5.2.2	Filtering and Distance Estimation	44
5.3	Evaluation of the Positioning Performance	46
5.3.1	Positioning Performance of a Moving Robot	46
5.3.2	Positioning Performance of a Pedestrian	54
6	Conclusion and Future Work	62
6.1	Conclusion	62
6.2	Future Work	63
A	Supplementary Materials	68
A.1	PDF Files	68
	References	69
	Literature	69
	Online sources	70

Acknowledgements

The completion of this master thesis would not have been possible without the guidance and support of my supervisors, FH-Prof. Dr.-Ing. habil. Hans-Georg Brachtendorf and Dr.-Ing. Christian Gentner, to whom I would like to express my deepest gratitude at this point. I would like to thank the German Aerospace Center for allowing me to participate in this project and for providing me with the resources. Furthermore, I would like to thank my friends and colleagues from Multimodal Navigation and Swarm Exploration research teams for the fun times we had working and socializing together. Finally, I must express my very profound gratitude to my family for the support along the way. This accomplishment would not have been possible without them.

Philipp Hager

Abstract

In recent years, indoor positioning using Bluetooth Low Energy (BLE) technology has gained significant attention as a consequence of its low power consumption, wide availability, and compatibility with a variety of devices. In this thesis, a BLE-based positioning system introducing a server-based BLE positioning method utilizing received signal strength (RSS) measurements is developed. Here, mobile devices, for instance smartphones, act as broadcasting BLE beacons. The BLE signal is received by static BLE sniffers installed in the environment. The sniffers obtain the RSS of the signal and send the measurements to a server. The position of the mobile device is calculated at the server using a particle filter (PF). As a countermeasure to fading effects caused by multipath propagation and to enhance the distance and position estimation accuracy, each sniffer is equipped with multiple closely spaced BLE receivers. To further increase the positioning accuracy, the RSS measurements are smoothed by a two-staged smoothing filter and a fusion of inertial data from the mobile device. The developed system is evaluated by indoor measurements, where a positioning error with a root mean square error (RMSE) of 0.75 m for a mobile device carried by a robot and a RMSE of 1.1 m for a walking pedestrian is obtained.

Kurzfassung

In den letzten Jahren hat die Positionierung in Innenräumen mittels der Bluetooth Low Energy (BLE)-Technologie aufgrund des geringen Stromverbrauchs, der breiten Verfügbarkeit und der Kompatibilität mit einer Vielzahl von Geräten große Aufmerksamkeit erlangt. In dieser Arbeit wird ein auf BLE beruhendes Positionierungssystem entwickelt, das eine serverbasierte BLE-Positionierungsmethode unter Verwendung von Messungen der Signalstärke realisiert. Dabei fungieren die mobilen Geräte, zum Beispiel Smartphones, als sendende BLE-Beacons. Das BLE-Signal wird von statischen, in der Umgebung installierten BLE-Sniffern empfangen. Die Sniffer erfassen die Empfangsstärke des Signals und senden die Messungen an einen Server. Die Position des mobilen Geräts wird auf dem Server mit Hilfe eines Particle Filters (PF) berechnet. Als Gegenmaßnahme zu den durch Mehrwegeausbreitung verursachten Fading-Effekten und zur Verbesserung der Genauigkeit der Entfernungs- und Positionsbestimmung ist jeder Empfänger mit mehreren eng nebeneinander platzierten BLE-Empfängern ausgestattet. Um die Positionierungsgenauigkeit weiter zu erhöhen, werden die Messungen der Signalstärke durch einen zweistufigen Glättungsfilter und eine Fusion von Inertialdaten des mobilen Geräts geglättet. Das entwickelte System wird durch Messungen in Innenräumen evaluiert, wobei ein mittlerer quadratischer Positionierungsfehler von 0,75 m für ein von einem Roboter getragenes mobiles Gerät und ein mittlerer quadratischer Positionierungsfehler von 1,1 m für einen Fußgänger erzielt wird.

Chapter 1

Introduction

1.1 Aim of this Thesis

Location-awareness is an essential part of modern life. In outdoor environments with clear sky view, this is done by using the global navigation satellite system (GNSS) technology. The GNSS technology is employed in many different applications, with in-car navigation systems, high-precision asset monitoring, and context-based services being just a few of the more popular ones. New use cases and application areas are also rapidly emerging.

The popularity of GNSS technology has created a long-standing demand for indoor positioning systems that are as accurate as those that serve outdoors. The GNSS provides localization precision in the cm range when used in combination with techniques like real-time kinematics (RTK). However, GNSS is ineffective for indoor positioning because of obstructions and multipath caused by walls, building structures, and objects; hence, alternate techniques are needed. The development of numerous standards and proprietary technologies such as radio frequency identification (RFID), ultra-wideband (UWB), Wi-Fi, Bluetooth, Bluetooth Low Energy (BLE), and camera-based positioning – each enabling a specific subset of use cases and applications – has been prompted by the need for equally ubiquitous, accurate, and dependable indoor positioning solutions for tracking and navigation [7].

Possible application scenarios for an indoor positioning system (IPS) include navigation in airports, health centers, museums, or department stores, but home automation applications are also conceivable. In addition to facilitating day-to-day indoor navigation, such a system could also help evacuate a building in the event of an emergency when visual aids become unusable.

The worldwide pandemic has pushed global development toward Industry 4.0 principles, including ubiquitous sensing, wireless communication, and cloud-based applications in all parts of the industrial sector and beyond. The rising need for precise control and monitoring of products was matched by rising demand for personal navigation solutions and proximity sensing to aid with social distance.

These advancements have highlighted the benefits of worldwide, standardized solutions for meeting indoor positioning needs and standards. Simplicity of deployment, operation, and maintenance have emerged as major design requirements for the adoption

of indoor positioning devices, where interoperability of functionality and components is also required for the components to work together as a system solution.

In the recent years, BLE has emerged as an important technology enabling indoor positioning and is chosen over other wireless technologies because of its low power consumption. Using wireless technologies like BLE, different approaches exist to determine an indoor location, each facing different benefits and problems. The goal of this thesis is the development of an IPS that relies on the signal strength of BLE signals. It is investigated, whether BLE is a suitable technology to enable precise indoor positioning and what further additional measures to enhance the positioning accuracy are necessary. The developed IPS is conceptualized, implemented, validated, and evaluated by performing indoor measurement campaigns, where the system is required to be highly stable and accurate, resource-efficient, and platform-independent.

1.2 Structure of this Thesis

This thesis is structured in six chapters, with the first introducing the motivation to start the work.

Chapter 2 covers the basic part of the work and deals with fundamentals regarding radio-based positioning. The first part focuses on the BLE standard, and the second part on the Bayesian estimation for non-linear systems.

In Chapter 3, the concept development and the planned methodological approach are introduced. The system architecture and the equipment in use are introduced. Furthermore, the data pre-processing and positioning algorithms are explained.

The practical implementation is focused on in Chapter 4, where details about the firmware and software implementation for the developed embedded IPS and an Android application are presented.

Chapter 5 describes the evaluation of the developed IPS using multiple indoor measurement campaigns with different setups. The algorithms are tested and evaluated.

Chapter 6 concludes the work with a summary and an outlook on potential further developments.

Chapter 2

Fundamentals of Radio-Based Positioning

In this chapter, the foundation for this thesis is built. The first section presents the trilateration positioning method and the BLE standard. The second section introduces the estimation theory for non-linear systems.

2.1 Radio-Based Positioning

As mentioned in Chapter 1, the field of indoor positioning has a wide range of applications and is therefore an important research topic. Smartphone-based indoor localization can be performed using a variety of approaches, including sensor-based relative navigation (i.e. dead reckoning), image-based navigation, and wireless localization [7]. Wireless technologies available on mobile devices include Wi-Fi, Bluetooth, BLE, UWB, and RFID. Because of its interoperability, low battery consumption, ease of deployment, and high range availability, BLE has been identified as the most appropriate wireless technology for use in the developed IPS for the objectives of this research.

2.1.1 Trilateration

Radio-based positioning schemes are most commonly used to estimate the unknown receiver's position by utilizing geometric relationships between transmitters and receivers relative to a known coordinate system. The coordinate system is established using N_c external static reference points, also referred to as anchors c_i , where the ground truth position of the anchors is known and the position of the target to be located B_b is unknown. The methods can be distinguished by the measurements used for estimating the position. In practice established methods are mostly range-based or angular-based systems. Range-based IPSs utilize the distances between the target B_b and the anchors c_i to estimate the position, collectively referred to as trilateration when three non-collinear reference points are used and referred to as multilateration, when more than three non-collinear anchors are part of the setup [2]. The principle of the trilateration is depicted in Figure 2.1.

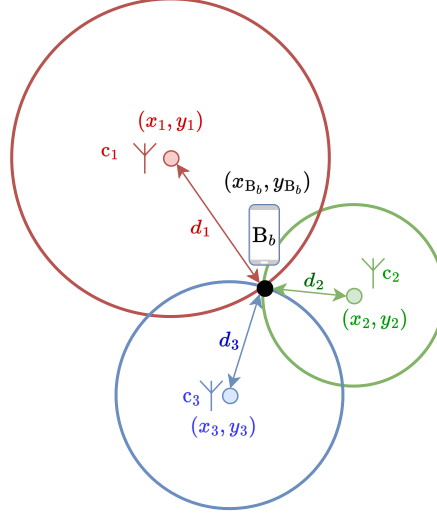


Figure 2.1: Principle of the trilateration method to estimate the position relative to a known coordinate system established by static anchors, adopted from [7].

In a two-dimensional (2D) scenario the position can be determined without ambiguities by establishing a linear equation system based on the distances between the target B_b and the anchors c_i , denoted as

$$(x_{B_b} - x_{c_i})^2 + (y_{B_b} - y_{c_i})^2 = d_i^2, \quad (2.1)$$

where d_i denotes the distance between the target B_b and the anchor c_i . The quadratic terms of the equation system can be eliminated by subtracting one equation from the equation system. The resulting linear equation system can be solved using the linear algebra method called ordinary least squares (OLS). The determination of the distance can be realized using a variety of approaches, for instance utilizing received signal strength (RSS) data or timing information such as Time of Arrival (ToA) or Time Difference of Arrival (TDoA). Location fingerprinting refers to other methodologies that use location-dependent properties of these measures rather than geometric relations [2]. In this thesis, the distance estimation is performed by measurements of the signal strength of the BLE wireless technology, which is described in the following section.

2.1.2 Bluetooth Low Energy (BLE)

The Bluetooth protocol [19] describes an international industry standard for wireless communication that was originally intended to replace serial data cables for connecting various devices. The development goals therefore included worldwide operation, low cost, robustness, short range, and low power consumption. To achieve these goals, Bluetooth operates in the 2.4 GHz industrial, scientific and medical (ISM) frequency band, which is license-free and can be used worldwide without requiring approval. For this reason, this frequency band is also used by other widely used standards (e.g. IEEE 802.11, IEEE 802.15.4, etc.) and is also susceptible to interference from other devices such as microwave ovens. To still provide stable connections, Bluetooth implements a

pseudo-random mechanism called adaptive frequency hopping (AFH). The Bluetooth specification was published in 1999 by the Bluetooth Special Interest Group, is continuously developed further, and is currently available in version 5.3. BLE was published in 2009 as part of the 4.0 specification but is to be regarded as independent of Bluetooth. The main focus of BLE is on connectivity and communication of devices in the ultra-low power segment. According to the physical layer of the BLE protocol stack, messages are sent via 40 channels, each 2 MHz wide, in the 2.4 GHz radio range. To conserve battery power, the BLE radio mostly operates in sleep mode and is only active while ongoing transmissions [19]. The spectrum of BLE is depicted in Figure 2.2.

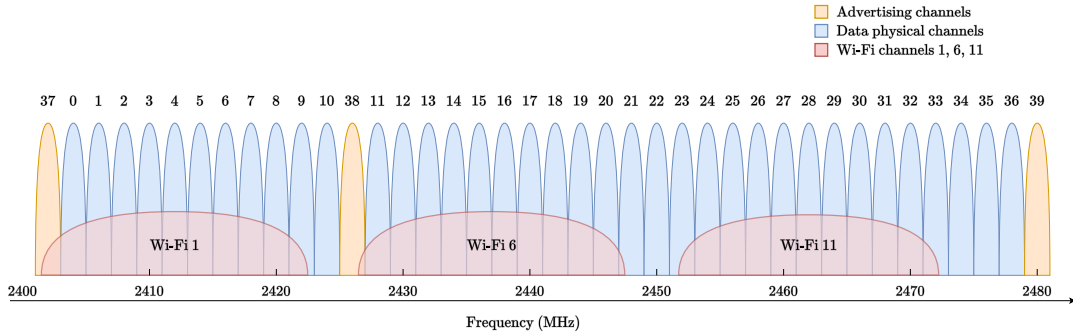


Figure 2.2: Physical layer of BLE spectrum dividing the 2.4 GHz ISM band into 40 channels each covering 2 MHz with overlapped Wi-Fi channels 1, 6 and 11, adopted from [8].

The link layer of the BLE protocol stack supports two different communication modes, namely connection-based and connection-less [19]. Connection-based mode refers to the packet-oriented, bidirectional, and encrypted transmission of large amounts of sensitive data between two already connected devices. Employing the connection-less mode, a packet-oriented, unidirectional, and unencrypted communication of non-sensitive data between non-connected devices is possible. This mode is therefore used to initiate a connection. The communication runs via three so-called advertisement channels 37, 38, and 39 provided for this purpose. The use of several advertisement channels increases the robustness against radio interference from other wireless communication technologies [19].

A BLE device that informs other BLE devices of its presence is referred to as an advertiser or beacon and periodically broadcasts so-called advertising events. An advertising event consists of a total of three advertising packets, each packet being broadcast on one of the three advertising channels. After each broadcast period, the advertiser waits for a potential response from another BLE device on the same channel before switching to the next advertising channel. The time between two consecutive advertising events is composed of the static advertising interval T_a and a pseudo-random delay γ . The static part T_a is a multiple of 0.625 ms between 20 ms and 10.485 s. The delay γ is between 0 ms and 10 ms and is intended to reduce the probability of collisions. BLE devices that detect advertisers are referred to as scanners. The three advertisement channels are scanned cyclically where parallel scanning is not possible. The scan window d_s denotes the pure detection time with active radio. The scan interval T_s determines

the time between two successive scan phases and the associated switch to the next advertising channel. An advertiser is detected by the scanner if it registers an advertising packet of the currently scanned advertising channel [19]. Figure 2.3 shows the timing of an advertiser and a scanner.

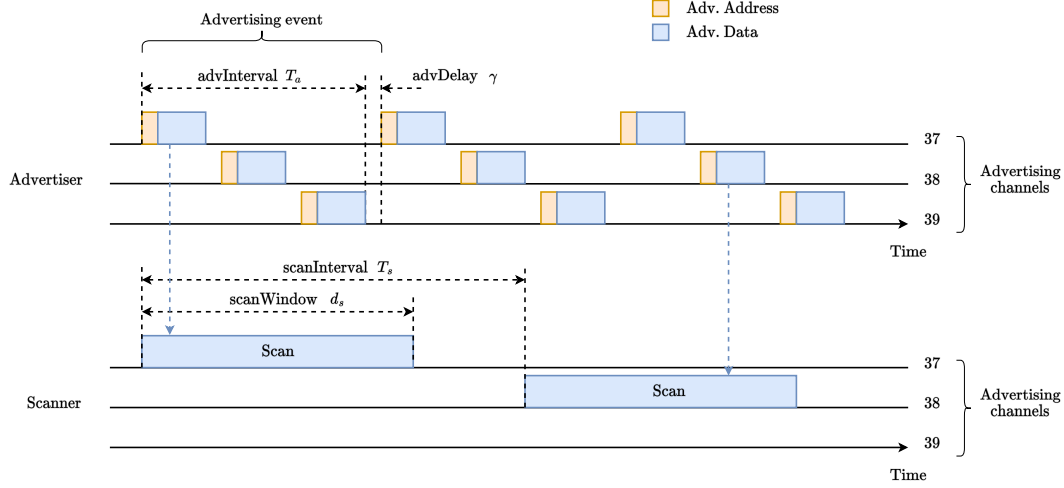


Figure 2.3: Principle of a BLE advertiser and scanner, where the advertiser is detected by the scanner when the channel that is currently being advertised overlaps with the currently scanned channel, adopted from [19].

During each advertising event, one advertising packet is sent per advertising channel. Figure 2.4 shows the predefined structure of an advertising packet, consisting of a preamble, a 4 byte access address, a protocol data unit (PDU), and a 3 byte cyclic redundancy check (CRC). The preamble is used for frequency synchronization in the receiver and consists of a 1 or 2 byte alternating sequence of bits, where the first bit of the preamble and the subsequent access address should match. The 4 byte access address is recommended to have the value 0x8E89BED6 according to the BLE protocol. The PDU of an advertising packet depends on the advertising type. As explained above, the advertising and scanning process only requires the scanner to receive the advertising packets and does not require a connection between the two devices to exchange further data. Therefore, it is sufficient to use undirected, unconnectable, and unscannable events. The suitable PDU type for this use case is referred to as `ADV_NONCONN_IND` [19].

The PDU consists of a 2 byte header and the advertising payload. The header provides information about the subsequent PDU type and the length of the subsequent payload. The advertising payload consists of the 6 byte advertising address and further partially customizable advertising data. The advertising address is used to identify the advertiser. The CRC is used to validate received packets. Some fields are reserved for future use (RFU) [19].

Furthermore, many BLE devices that scan the advertising channels for advertising packets are not only capable of receiving the packets, but their BLE radios also offer a measurement of the RSS which is often available as so-called received signal strength indicator (RSSI) in the unit dBm. The RSS depends among other aspects on the distance

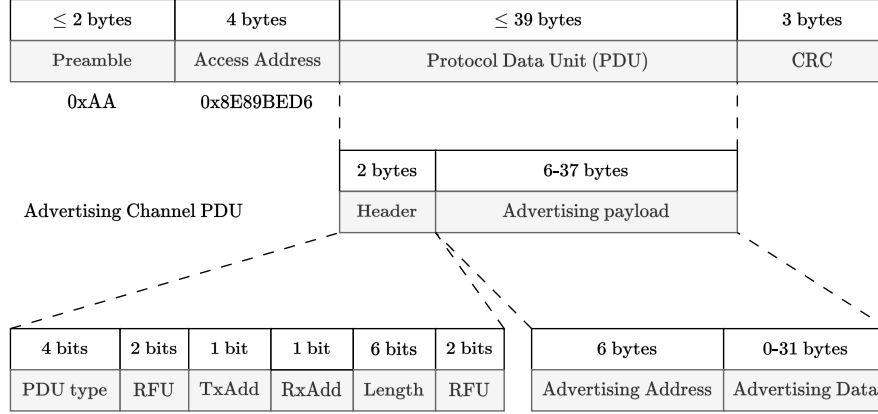


Figure 2.4: Structure of a BLE advertising packet split up into individual parts, adopted from [19].

between a transmitter and a receiver [9]. This relation is used for distance determination described in Section 2.1.1 and thus forms the base for the indoor localization approach in this thesis.

2.1.3 Path Loss and Multipath Propagation

In this thesis, the position is estimated using RSSI data, which is a dBm measurement of the received power in the observed frequency channel and is given by

$$\text{RSSI} = 10 \cdot \log_{10} \left(\frac{P_r}{1 \text{ mW}} \right). \quad (2.2)$$

The RSSI represents the received power P_r referenced to 1 mW [12].

According to [9], a wireless signal traveling the distance d between a transmitter and a receiver in non-open space will be received with a power of

$$P_r = P_t G_t G_r \left(\frac{\lambda}{4\pi d} \right)^\gamma. \quad (2.3)$$

In Equation (2.3), $\lambda = c_0/f$ denotes the wavelength of the signal, c_0 is the speed of light, f is the frequency of the signal, γ is an environment-dependent path loss exponent, d denotes the distance between the transmitter and the receiver and G_t and G_r are the transmitter and receiver gains, which are known or can be calibrated, respectively. With $\gamma = 2$, Equation (2.3) is referred to as Friis Free-Space-Model. In the logarithmic domain, Equation (2.3) can be expressed as

$$\underbrace{\text{RSSI}(d)}_{\text{dBm}} = \underbrace{P_t}_{\text{dBm}} + \underbrace{G_t}_{\text{dBi}} + \underbrace{G_r}_{\text{dBi}} + \underbrace{K}_{\text{dBm}} - \gamma \cdot 10 \cdot \log_{10}(d), \quad (2.4)$$

where constant factors of Equation (2.3) are represented by the variable K . With $\beta_0 = P_t + G_t + G_r + K$ and $\beta_1 = -\gamma \cdot 10$, Equation (2.4) can be rewritten in a compact form as

$$\text{RSSI}(d) = \beta_0 + \beta_1 \cdot \log_{10}(d), \quad (2.5)$$

which is collectively referred to as the log-distance path loss model [12].

The propagation of a radio signal emitted by a transmitter is strongly influenced by the environment. Radio waves traversing a medium are influenced by phenomena such as diffraction, reflection, and scattering before reaching the receiver, as depicted in Figure 2.5. Reflections occur when a signal strikes a smooth object bigger than the signal's wavelength, such as large smooth walls. Scattering happens when a signal collides with an item with irregular surface properties smaller than the wavelength of the signal, resulting in a split of the signal into multiple partial signals. Diffraction occurs when a signal strikes an object with sharp edges, such as the edges of obstacles and walls, resulting in a shift in signal direction. In addition to the power loss, these occurrences force the signal to travel over several pathways besides the line of sight (LOS) to the receiver, resulting in so-called multipath fading. Due to different path lengths, these pathways have variable delays and attenuations, causing the transmitted signal to smear, resulting in severe issues with RSS-based positioning, particularly indoors [12].

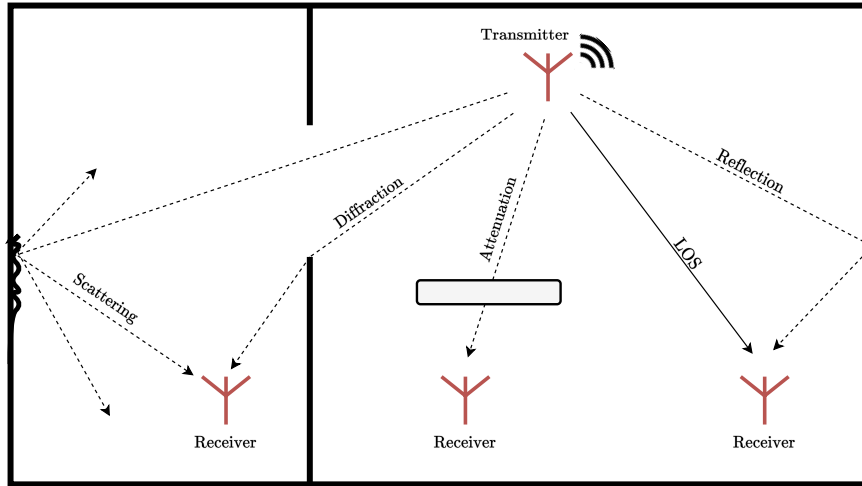


Figure 2.5: Multipath phenomena in an indoor environment showing reflection, attenuation, diffraction, and scattering, adopted from [12].

Compared to other methodologies, RSS-based positioning systems are relatively cheap and easy to implement, rely on low-complexity algorithms and do not require time synchronization. It has to be kept in mind, that determination of relevant parameters for the path loss model is necessary and the propagation is heavily influenced by the surroundings.

2.2 State Estimation of Non-Linear Systems

The purpose of this thesis is the development of an IPS for mobile devices, where positioning describes the process of finding a platform's spatial location with respect to a known coordinate system. The fundamentals of position estimation are discussed in the following sections.

2.2.1 Introduction to Recursive Bayesian Filtering

Positioning aims to estimate the kinematic state, i.e. the position, velocity, and orientation, of the target under investigation based on a so-called state-space approach. Often the state of a system is not directly observable but has to be estimated based on measurements, also called observations. A widely used and in practice established estimation technique is the Bayesian Filter, which tries to perform an optimal state estimation by recursively (sequentially) calculating the probability density function (PDF) of the state [1]. The relevant information for describing the system is summarized in a state vector $\mathbf{x}(t_k)$ at time instant t_k . A general dynamic system can be mathematically modeled by a discrete-time state-space description consisting of two equations.

The transition model,

$$\mathbf{x}(t_k) = \mathbf{f}(\mathbf{x}(t_{k-1}), \mathbf{u}(t_k), \mathbf{n}_t(t_k)) , \quad (2.6)$$

describes the evolution of the previous state $\mathbf{x}(t_{k-1})$ at time unit t_{k-1} to the current state $\mathbf{x}(t_k)$ at time t_k using the possible non-linear function $\mathbf{f}(\cdot, \cdot, \cdot)$. Control data, such as additional sensor data, is represented by $\mathbf{u}(t_k)$ while the process noise is denoted by $\mathbf{n}_t(t_k)$. From a probabilistic perspective, the transition model in Equation (2.6) can be represented compactly by the conditional PDF $p(\mathbf{x}(t_k) | \mathbf{x}(t_{k-1}))$, also referred to as transition prior.

The measurement model,

$$\mathbf{z}(t_k) = \mathbf{h}(\mathbf{x}(t_k), \mathbf{n}_h(t_k)) , \quad (2.7)$$

describes the relation between the measurements $\mathbf{z}(t_k)$ and the current state $\mathbf{x}(t_k)$ using the possible non-linear function $\mathbf{h}(\cdot, \cdot)$ with the measurement noise $\mathbf{n}_h(t_k)$. Similarly, the measurement model can also be expressed as likelihood using a conditional PDF as $p(\mathbf{z}(t_k) | \mathbf{x}(t_k))$ [5].

Many modern dynamic systems have high requirements for real-time capability, due to their architecture as multi-sensor networks. The evolution of the target state $\mathbf{x}(t_k)$ is calculated sequentially when a new measurement is available based only on the previous state so it is not mandatory to keep the entire history and calculate it repeatedly at each time step, noted as

$$p(\mathbf{x}(t_k) | \mathbf{x}(t_{1:k-1})) = p(\mathbf{x}(t_k) | \mathbf{x}(t_{k-1})) . \quad (2.8)$$

Prior observations $\mathbf{z}(t_{1:k-1})$ have no impact on the evolution of the state $\mathbf{x}(t_k)$, since they were already considered in the previous state $\mathbf{x}(t_{k-1})$, which is noted as

$$p(\mathbf{x}(t_k) | \mathbf{x}(t_{k-1}), \mathbf{z}(t_{k-1})) = p(\mathbf{x}(t_k) | \mathbf{x}(t_{k-1})) . \quad (2.9)$$

This process is referred to as recursive estimation and is the basic idea of the so-called first order hidden Markov model (HMM). The states are hidden and can only be observed by measurements [4]. The principle of a first order HMM in combination with the transition and measurement model is depicted in Figure 2.6.

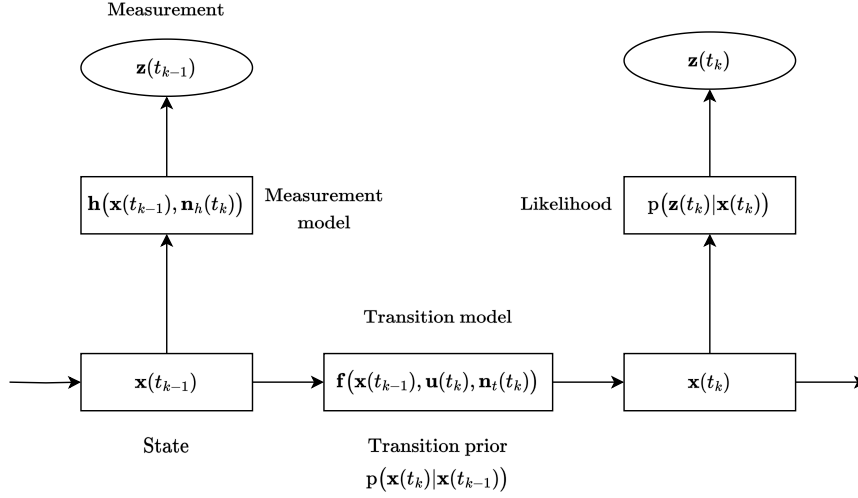


Figure 2.6: Principle of a first order HMM where the current state $\mathbf{x}(t_k)$ is observed using a measurement $\mathbf{z}(t_k)$ and only depends on the previous state $\mathbf{x}(t_{k-1})$, adopted from [4].

In the Bayesian framework, the construction of the PDF $p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k}))$ of the state given all prior observations $\mathbf{z}(t_{1:k-1})$ is performed in two stages: the prediction and the update [16].

The prediction stage predicts an a priori distribution $p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k-1}))$ based on the old estimate $p(\mathbf{x}(t_{k-1}) | \mathbf{z}(t_{1:k-1}))$ by

$$p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k-1})) = \int p(\mathbf{x}(t_k) | \mathbf{x}(t_{k-1})) \cdot p(\mathbf{x}(t_{k-1}) | \mathbf{z}(t_{1:k-1})) d\mathbf{x}(t_{k-1}) . \quad (2.10)$$

The update stage updates the state distribution to form an a posteriori estimate $p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k}))$ by considering the newly available measurement $\mathbf{z}(t_k)$ using Bayes' rule by

$$\begin{aligned} p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k})) &= p(\mathbf{x}(t_k) | \mathbf{z}(t_k), \mathbf{z}(t_{1:k-1})) \\ &= \frac{p(\mathbf{z}(t_k) | \mathbf{x}(t_k), \mathbf{z}(t_{1:k-1})) \cdot p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k-1}))}{p(\mathbf{z}(t_k) | \mathbf{z}(t_{1:k-1}))} \\ &= \frac{p(\mathbf{z}(t_k) | \mathbf{x}(t_k)) \cdot p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k-1}))}{p(\mathbf{z}(t_k) | \mathbf{z}(t_{1:k-1}))} , \end{aligned} \quad (2.11)$$

with the normalization constant

$$p(\mathbf{z}(t_k) | \mathbf{z}(t_{1:k-1})) = \int p(\mathbf{z}(t_k) | \mathbf{x}(t_k)) \cdot p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k-1})) d\mathbf{x}(t_k) . \quad (2.12)$$

Equation (2.10) describes the prediction of the state, whereas Equation (2.11) expresses the impact of observations on the state. With both Equations (2.10) and (2.11), the general discrete-time filter problem in the sense of Bayes is formally described. Due to the limited possibility of representing a continuous probability distribution, Bayesian

filters represent only a conceptual solution, which in general cannot be determined analytically [1]. However, there are methods to approach filtering of non-linear, non-Gaussian, dynamic systems, for instance numerical approaches, such as the so-called particle filter (PF), which is focused on in the following section.

2.2.2 Particle Filter Basics

Assuming linear systems and observations with Gaussian noise, the Kalman filter (KF) is the optimal algorithm for the recursive problem. However, many practical problems are more complex and are subject to strongly non-linear characteristics and non-Gaussian noise, making the assumptions of a KF invalid. The need for accurate recursive state estimation methods of non-linear problems led to the development of the Extended Kalman filter (EKF) and the Unscented Kalman Filter (UKF). The EKF performs the estimation based on the linearization of the system model by a first-order Taylor series at the point of the mean value of the distribution. For small sampling intervals, the required linearity of the system is therefore no longer necessary. In contrast to the analytical approximation of the EKF, the UKF estimates the nonlinearity by a statistical approximation. However, for strongly non-linear systems, the EKF and UKF show bad performance. The development of computing power in recent years has made computationally intensive statistical procedures practical. The estimation problem is directly resolved using Monte Carlo techniques as opposed to estimating the system via linearization. One possible implementation of a recursive Bayesian filter using Monte Carlo methods is the PF [16].

The fundamental concept is to directly approximate the a posteriori distribution $p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k}))$ by using a finite set S containing N_p state samples $\mathbf{x}^{(j)}(t_k)$, collectively referred to as particles, with associated relative weights $w^{(j)}(t_k)$, denoted as

$$S = \left\{ \mathbf{x}^{(j)}(t_k), w^{(j)}(t_k) \right\}_{j=1}^{N_p} . \quad (2.13)$$

The state of a particle $\mathbf{x}^{(j)}(t_k)$ can be updated at each time step by the transition model with Equation (2.6). The a posteriori distribution $p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k}))$ can then be approximated with

$$p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k})) \approx \frac{1}{N_p} \sum_{j=1}^{N_p} w^{(j)}(t_k) \cdot \delta(\mathbf{x}(t_k) - \mathbf{x}^{(j)}(t_k)) , \quad (2.14)$$

where $\delta(\cdot)$ represents the Dirac-Impulse. For $N_p \rightarrow \infty$, the approximation error vanishes and the particle representation corresponds to the continuous density function. Equation (2.14) is then no longer only an approximation but an equivalent description [16].

Monte-Carlo-Integration

According to Equation (2.11), the a posteriori distribution can be estimated by recursively solving a multidimensional integral. Using Monte-Carlo-Integration [16], a multi-dimensional integral of a function $f(\mathbf{x})$ and a PDF $p(\mathbf{x})$, the expectation of $f(\mathbf{x})$ can be

approximated by

$$F = \int f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \approx \frac{1}{N_p} \sum_{j=1}^{N_p} f(\mathbf{x}^{(j)}) , \quad (2.15)$$

using a sufficient number of samples N_p .

Since the PDF might be a complex, high-dimensional distribution, drawing samples from it is not always practical. By drawing the particles in accordance with the idea of importance sampling, this issue can be resolved. This idea draws the particles not in accordance with the PDF $p(\mathbf{x})$, but rather in accordance with a so-called significance function $q(\mathbf{x})$, from which samples might be taken. Equation (2.15) can then be denoted as

$$F = \int f(\mathbf{x}) q(\mathbf{x}) \underbrace{\frac{p(\mathbf{x})}{q(\mathbf{x})}}_{w^*(\mathbf{x})} d\mathbf{x} = \int f(\mathbf{x}) q(\mathbf{x}) w^*(\mathbf{x}) d\mathbf{x} \approx \frac{1}{N_p} \sum_{j=1}^{N_p} f(\mathbf{x}^{(j)}) \cdot w^*(\mathbf{x}^{(j)}) \quad (2.16)$$

with

$$w^*(\mathbf{x}^{(j)}) = \frac{p(\mathbf{x}^{(j)})}{q(\mathbf{x}^{(j)})} . \quad (2.17)$$

With Equation (2.16), the PDF $p(\mathbf{x})$ can be approximated by drawing N_p random samples $\mathbf{x}^{(j)}$ of the PDF $q(\mathbf{x})$, where each sample has a weight $w^*(\mathbf{x}^{(j)})$ [16]. The weights $w^*(\mathbf{x}^{(j)})$ are normalized to guarantee a sum of unity by

$$w(\mathbf{x}^{(j)}) = \frac{w^*(\mathbf{x}^{(j)})}{\sum_{j=1}^{N_p} w^*(\mathbf{x}^{(j)})} . \quad (2.18)$$

To sample the particles and to determine the weights, an importance function $q(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k}))$ is necessary. If the importance function is chosen that

$$q(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k})) = q(\mathbf{x}(t_k) | \mathbf{x}(t_{k-1}), \mathbf{z}(t_k)) \cdot q(\mathbf{x}(t_{k-1}) | \mathbf{z}(t_{1:k-1})) , \quad (2.19)$$

the weighting can be updated recursively. The derivation of the weight update is done by reformulating the recursive update step in Equation (2.19) to

$$\begin{aligned} p(\mathbf{x}^{(j)}(t_k) | \mathbf{z}(t_{1:k})) &= \frac{p(\mathbf{z}(t_k) | \mathbf{x}^{(j)}(t_k)) \cdot p(\mathbf{x}^{(j)}(t_k) | \mathbf{z}(t_{1:k-1}))}{p(\mathbf{z}(t_k) | \mathbf{z}(t_{1:k-1}))} \\ &= \frac{p(\mathbf{z}(t_k) | \mathbf{x}^{(j)}(t_k)) \cdot p(\mathbf{x}^{(j)}(t_k) | \mathbf{z}(t_{k-1}))}{p(\mathbf{z}(t_k) | \mathbf{z}(t_{1:k-1}))} \cdot p(\mathbf{x}^{(j)}(t_{k-1}) | \mathbf{z}(t_{1:k-1})) \\ &\approx p(\mathbf{z}(t_k) | \mathbf{x}^{(j)}(t_k)) \cdot p(\mathbf{x}^{(j)}(t_k) | \mathbf{x}^{(j)}(t_{k-1})) \\ &\quad \cdot p(\mathbf{x}^{(j)}(t_{k-1}) | \mathbf{z}(t_{1:k-1})) . \end{aligned} \quad (2.20)$$

The recursive weight update is calculated by inserting the Equations 2.19 and 2.20 in Equation 2.17 resulting in

$$\begin{aligned} w^{*(j)}(t_k) &\approx \frac{p(\mathbf{z}(t_k) \mid \mathbf{x}^{(j)}(t_k)) \cdot p(\mathbf{x}^{(j)}(t_k) \mid \mathbf{x}^{(j)}(t_{k-1}))}{q(\mathbf{x}^{(j)}(t_k) \mid \mathbf{x}^{(j)}(t_{1:k-1}), \mathbf{z}(t_{1:k}))} \cdot \frac{p(\mathbf{x}^{(j)}(t_{k-1}) \mid \mathbf{z}(t_{1:k-1}))}{q(\mathbf{x}^{(j)}(t_{k-1}) \mid \mathbf{z}(t_{1:k-1}))} \\ &= w^{(j)}(t_{k-1}) \frac{p(\mathbf{z}(t_k) \mid \mathbf{x}^{(j)}(t_k)) \cdot p(\mathbf{x}^{(j)}(t_k) \mid \mathbf{x}^{(j)}(t_{k-1}))}{q(\mathbf{x}^{(j)}(t_k) \mid \mathbf{x}^{(j)}(t_{1:k-1}), \mathbf{z}(t_{1:k}))} \end{aligned} \quad (2.21)$$

with

$$w^{(j)}(t_k) = \frac{w^{*(j)}(t_k)}{\sum_{j=1}^{N_p} w^{*(j)}(t_k)} . \quad (2.22)$$

Sequential Importance Sampling Particle Filter

Equation (2.21) forms the basis for the so-called sequential importance sampling particle filter (SISPF), on which further types of PFs are based [16]. A SISPF is subject to a problem referred to as degeneracy which leads to divergence over time, meaning after a few iterations, all but one particle will have negligible weight and thus do not contribute to the posterior distribution. The phenomenon of degeneracy can be approximately quantified by calculating the effective sample size N_{eff} , defined as

$$N_{\text{eff}} \approx \frac{1}{\sum_{j=1}^{N_p} \left(w^{(j)}(t_k) \right)^2} . \quad (2.23)$$

The phenomenon of degeneracy is an undesirable effect in PFs but can be counteracted by either increasing the number of particles N_p or choosing the importance density

$$q(\mathbf{x}^{(j)}(t_k) \mid \mathbf{x}^{(j)}(t_{k-1}), \mathbf{z}(t_k)) = p(\mathbf{x}^{(j)}(t_k) \mid \mathbf{x}^{(j)}(t_{k-1})) \quad (2.24)$$

simplifying the recursive weight update in Equation 2.21 to

$$w^{*(j)}(t_k) = w^{(j)}(t_{k-1}) \cdot p(\mathbf{z}(t_k) \mid \mathbf{x}^{(j)}(t_k)) . \quad (2.25)$$

Furthermore, the concept of resampling can be deployed, which is the basic idea of the so-called sequential importance resampling particle filter (SIRPF) [16].

2.2.3 Resampling

Resampling is one possible countermeasure to degeneracy [16]. With a resampling algorithm, particles with a high weight survive and will be duplicated while particles with low weights are eliminated, so the number of particles stays constant. The resampling is done by transforming the particle set S to S^* by

$$S = \left\{ \mathbf{x}^{(j)}(t_k), w^{(j)}(t_k) \right\}_{j=1}^{N_p} \xrightarrow{\text{Resampling}} S^* = \left\{ \mathbf{x}^{(j)*}(t_k), w^{(j)*}(t_k) \right\}_{j=1}^{N_p} . \quad (2.26)$$

The resampling can be done in various ways, for instance by systematic resampling [6]. The resampled set S^* is constructed by sampling N_p times from a discrete version of the posterior density $p(\mathbf{x}(t_k) \mid \mathbf{z}(t_{1:k}))$ everytime the resampling is performed.

The first step of the systematic resampling involves the construction of the cumulative distribution function (CDF) $\mathbf{c}$ by summing up the normalized weights $w^{(j)}(t_k)$. After that, a random discrete starting point u_1 is drawn from the uniform distribution $\mathcal{U}[0, N_p^{-1}]$. The remaining discrete subpartitions u_i are constructed by $u_j = u_1 + N_p^{-1} \cdot (j - 1)$ with $j \in 1, \dots, N_p$. Systematic resampling works by comparing the CDF $\mathbf{c}$ to the discrete subpartitions u_j . The number of times the discrete subpartition u_j is within the range of $(c_{i-1}, c_i]$ determines how often the j -th particle $\mathbf{x}^{(j)}(t_k)$ is resampled (replicated). The new weights $w^{(j)*}(t_k)$ are all set to uniform weight $w^{(j)*}(t_k) = N_p^{-1}$ after the resampling procedure. In the example of Figure 2.7 the CDF $\mathbf{c}$ is represented by the blue solid line. The particle $\mathbf{x}^{(3)}(t_k)$ is reproduced twice, since u_3 and u_4 fall into the weight $w^{(3)}(t_k)$. If two or more u_j fall into the same step height, the particle of this step height is reproduced accordingly. In Algorithm 2.1 a pseudo-code of the systematic resampling algorithm is shown. The CDF is presented by a vector $\mathbf{c}$.

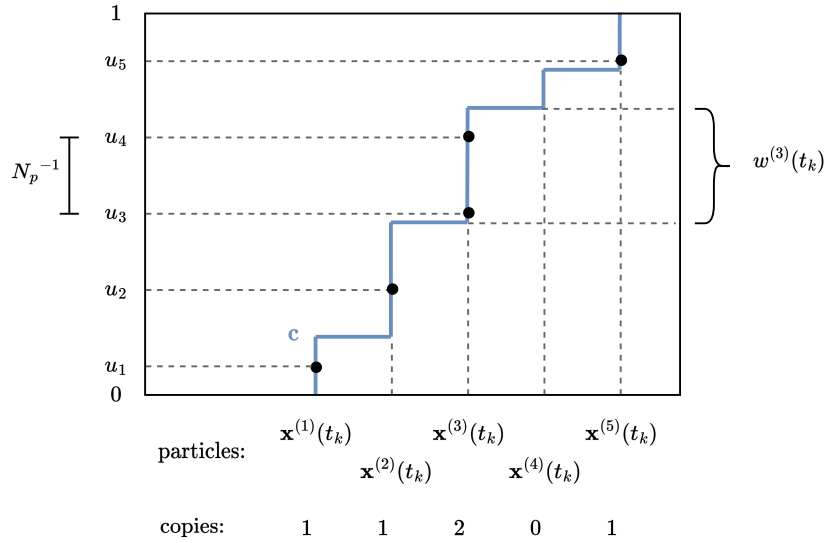


Figure 2.7: Systematic resampling used as a measure against degeneracy in PFs, adopted from [6].

Algorithm 2.1: Systematic Resampling

Input: $\left[\left\{ \mathbf{x}^{(j)}(t_k), w^{(j)}(t_k) \right\}_{j=1}^{N_p} \right]$

Output: $\left[\left\{ \mathbf{x}^{*(j)}(t_k), w^{*(j)}(t_k) \right\}_{j=1}^{N_p} \right]$

- 1: Initialize CDF: $c_1 \leftarrow w^{(1)}(t_k)$
- 2: **for** $j \leftarrow 2$ to N_p **do**
- 3: Construct CDF: $c_j \leftarrow c_{j-1} + w^{(j)}(t_k)$
- 4: Start at bottom of CDF: $i \leftarrow 1$
- 5: Draw starting point: $u_1 \sim \mathcal{U} \left[0, N_p^{-1} \right]$
- 6: **for** $j \leftarrow 1$ to N_p **do**
- 7: Move along CDF: $u_j \leftarrow u_1 + N_p^{-1}(j-1)$
- 8: **while** $u_j > c_i$ **do**
- 9: $i \leftarrow i + 1$
- 10: Assign particle: $\mathbf{x}^{*(j)}(t_k) \leftarrow \mathbf{x}^{(i)}(t_k)$
- 11: Assign weight: $w^{*(j)}(t_k) \leftarrow N_p^{-1}$

2.2.4 Sampling Importance Resampling Particle Filter

The extension of a SISPF to include the concept of resampling is referred to as SIRPF [16], described in Algorithm 2.2. The SIRPF selects the significance function to be equal to the previous distribution $p(\mathbf{x}^{(j)}(t_k) | \mathbf{x}^{(j)}(t_{k-1}))$ to create particles. The weight update in Equation (2.25) is simplified as a result of the resampling carried out after each iteration to

$$w^{*(j)}(t_k) = p(\mathbf{z}(t_k) | \mathbf{x}^{(j)}(t_k)) . \quad (2.27)$$

Algorithm 2.2: Sequential Importance Resampling Particle Filter

Input: $\left[\left\{ \mathbf{x}^{(j)}(t_{k-1}), w^{(j)}(t_{k-1}) \right\}_{j=1}^{N_p}, \mathbf{z}(t_k) \right]$

Output: $\left[\left\{ \mathbf{x}^{(j)}(t_k), w^{(j)}(t_k) \right\}_{j=1}^{N_p} \right]$

- 1: $\mathbf{x}^{(j)}(t_k) \leftarrow f(\mathbf{x}^{(j)}(t_{k-1}), \mathbf{u}(t_{k-1}))$
- 2: **for** $j \leftarrow 1$ to N_p **do**
- 3: Draw $\mathbf{x}^{(j)}(t_k) \sim p(\mathbf{x}^{(j)}(t_k) | \mathbf{x}^{(j)}(t_{k-1}))$
- 4: Calculate $w^{*(j)}(t_k) \leftarrow p(\mathbf{z}(t_k) | \mathbf{x}^{(j)}(t_k))$
- 5: Calculate total weight: $W \leftarrow \sum_{j=1}^{N_p} w^{(j)}(t_k)$
- 6: **for** $j \leftarrow 1$ to N_p **do**
- 7: Normalize: $w^{(j)}(t_k) \leftarrow W^{-1} w^{*(j)}(t_k)$
- 8: $\left\{ \mathbf{x}^{(j)}(t_k), w^{(j)}(t_k) \right\}_{j=1}^{N_p} \leftarrow \text{Resampling} \left(\left\{ \mathbf{x}^{(j)}(t_k), w^{(j)}(t_k) \right\}_{j=1}^{N_p} \right)$

The procedure of one iteration of a SIRPF including all steps is illustrated in Figure 2.8. The weights $w^{(j)}(t_k)$ are calculated based on the measurements $\mathbf{z}(t_k)$ and the conditional probability $p(\mathbf{z}(t_k) | \mathbf{x}(t_k))$ indicated by the black solid line. Then the particles are resampled, where high-weighted ones are reproduced and low-weighted ones are eliminated, so the number of particles remains constant. After the resampling, all particles have a uniform weight N_p^{-1} . Based on the transition model and the previous state $\mathbf{x}^{(j)}(t_{k-1})$ of the particles, the next state $\mathbf{x}^{(j)}(t_k)$ is finally predicted.

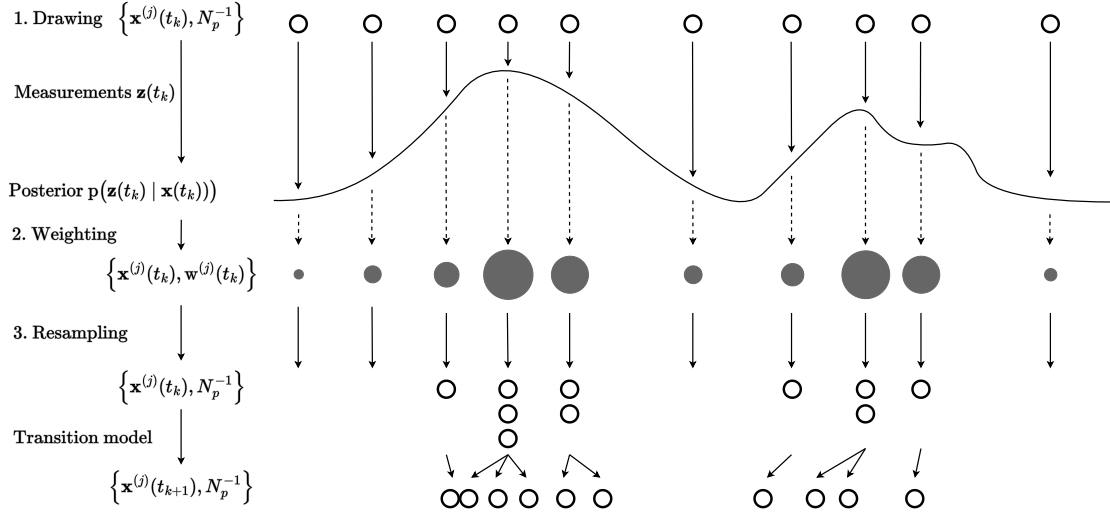


Figure 2.8: One iteration of SIRPF including the steps: Drawing, weighting, and resampling, adopted from [16].

However, resampling solves the problem of degeneracy but introduces other problems. The opportunity for parallelization is limited. Furthermore, the high-weighted particles are multiplied often, which leads to a loss of diversity and the particles concentrate in a single point [1].

Target State Estimation

In many practical applications like the estimation of a distinct position, a point estimate is more useful than a distribution. To determine a point estimate based on the posterior filtered distribution, different strategies exist. The point estimate can either be calculated with the minimum mean square error (MMSE) following the formula

$$\mathbf{x}_{\text{MMSE}}(t_k) = \int \mathbf{x}(t_k) \cdot p(\mathbf{x}(t_k) | \mathbf{z}(t_{1:k})) \, d\mathbf{x}(t_k) \approx \sum_{j=1}^{N_p} w^{(j)}(t_k) \mathbf{x}^{(j)}(t_k), \quad (2.28)$$

which also represents the expected value of the posterior PDF. Alternatively, the maximum a posteriori (MAP) estimate could be determined by using the highest weighted particle [5].

Chapter 3

Concept of a BLE-Based Indoor Positioning System

In the following sections, the IPS is fully conceptualized. The problem is considered from a hardware and user perspective, and a justified system architecture is proposed based on this. A detailed description of the measurement setup and resulting considerations regarding a pre-processing of the data is given.

3.1 System Architecture of an Indoor Positioning System

One key feature of modern smartphones is the availability of BLE, which can for instance be used for home automation purposes or communication with personal fitness monitoring sensors to name a few options. As stated in Section 2.1.3, BLE can also be used for positioning purposes utilizing distance estimations similar to the fundamental principle GNSS positioning is based on.

An IPS for mobile devices can be realized using two different approaches. The position can be determined by the mobile device itself, also referred to as device-based approach. In that case, the mobile device acts as a receiver and static anchors act as transmitters. The distance estimation to the anchors and the resulting positioning estimation is performed by the mobile device itself. Another possible implementation is a server-based approach, where the mobile device acts as a transmitter, also referred to as beacon. The static anchors act as receivers and forward the received signal emitted by the smartphone to a server. The server performs the distance and position estimation.

Using a customary smartphone with an Android operating system, the access to BLE advertising and scanning is limited. The parameter values T_a , T_s , and d_s of advertising and scanning procedure cannot be directly chosen. Instead, only predefined values are available for selection. Effects like multipath propagation must be taken into account when estimating the distance to get the highest level of accuracy. Modern BLE distance estimate algorithms therefore average the RSSIs over all channels before estimating the distance. The accuracy of distance calculation can be greatly increased by calculating the mean across all channels. The averaging is necessary due to channel dependent RSSI, which is investigated in [10]. As mentioned in Section 2.1.2, BLE uses three different channels for advertising. The center frequencies of the advertising channels are located at

2.402 GHz, 2.426 GHz and 2.480 GHz and thus entirely spread across the used frequency band. Reasoning for the necessity of an averaging are frequency dependent antenna gains G_r and G_t , frequency dependent path loss as can be seen in Equation (2.3) and different multipath effects, described in Section 2.1.3. However, in many practical scenarios, the mean over all channels cannot be computed. The aforementioned scan window on various Android smartphones lasts for 4.096 s. The smartphone and its surroundings would have to remain still for three times 4.096 s to determine the mean across all BLE channels, which is impractical for dynamic scenarios [10]. Moreover, the BLE channel information is not available on Android smartphones without additional measures. However, the BLE channel can be determined, as shown in [10], which is particularly useful when frequency dependent error mitigation is considered. Considering the large scanning intervals of 4.096 s on Android smartphones, the limited access to radio settings and the limited computation power on mobile devices, a device-based approach is not expected to lead to the desired functionality of a practically usable IPS. The IPS focused on in this thesis is therefore planned to be implemented using a server-based approach, consisting of a mobile device acting as a BLE beacon and multiple static BLE receivers, as illustrated in Figure 3.1.

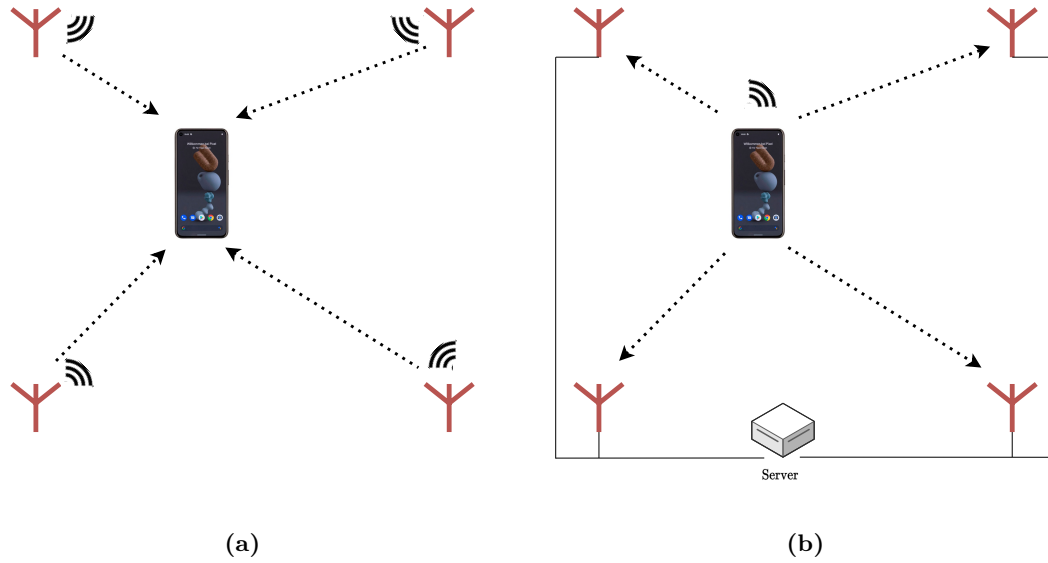


Figure 3.1: Comparison of two possible system architectures implemented as device-based and server-based IPS.

(a) Device-based approach, consisting of multiple static broadcasting BLE beacons and a receiving mobile device, that determines its position on its own.

(b) Server-based approach, consisting of multiple static BLE receivers connected to a server and a broadcasting mobile device acting as a BLE beacon, where the position is determined on the server.

This server-based method is not limited by smartphone manufacturer's Application Programming Interface (API) constraints. Furthermore, the BLE receivers can provide extra information besides the RSSI, for instance channel information.

3.2 Hardware and Software Platforms

The hardware components used in the experiments are selected based on their technical specifications. As BLE receiver, the widely used and low-cost 2.4 GHz transceiver nRF52840 Dongle from Nordic Semiconductor [18] is chosen. The nRF52840 Dongle features a printed circuit board (PCB) antenna, a powerful ARM Cortex M4 with floating point support and fully supports BLE v5. The nRF52840 Dongle supports serial port communication over Universal Serial Bus (USB) allowing further processing on a host machine providing a serial port interface. The nRF52840 Dongle is depicted in Figure 3.2.

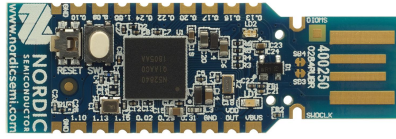


Figure 3.2: nRF52840 Dongle [18] used as a BLE receiver to record BLE packets advertised from a mobile device acting as BLE beacon.

Based on the hardware platform, the integrated development environments Segger Embedded Studio and nRF Connect for Desktop are selected for programming and debugging the used hardware based on embedded C programming. Additionally the BLE software stack S140-SoftDevice-v5.0.1 provided by Nordic is used. For prototyping, curve fitting, data plotting and algorithm development Matlab is utilized. As mobile device, the Android smartphone Google Pixel 5 is used. The Android application for the mobile device is developed using Android Studio.

3.3 Determination of the Path Loss Model

To acquire a better understanding of the BLE signal behavior under indoor conditions, a first indoor measurement campaign is conducted. The indoor measurements are performed in a $14\text{ m} \times 8\text{ m}$ radio measurements laboratory of the Institute of Communications and Navigation at the German Aerospace Center (DLR). The ceiling of the laboratory is equipped with a Vicon¹ high-precision optical motion capture system consisting of 20 infrared cameras and 20 infrared strobes. This setup can locate objects in the observation area with a localization error of less than 1 cm and provides a ground truth reference for all conducted measurements.

In the first step, the determination of the path loss model involves a static BLE receiver and a moving smartphone acting as a BLE beacon. The smartphone is mounted on a robot which is programmed to move along a defined path S_1 with intermediate standing phases while the BLE receiver is statically mounted on a tripod. Figure 3.3a shows the indoor environment with the robot and the BLE receiver $c_{4,1}$. Figure 3.3b shows the ground truth view of Figure 3.3a from the point of view of the optical motion

¹<https://www.vicon.com>

capture system where the path S_1 of the robot is shown with the black arrow and the BLE receiver $c_{4,1}$ is shown in red (—).

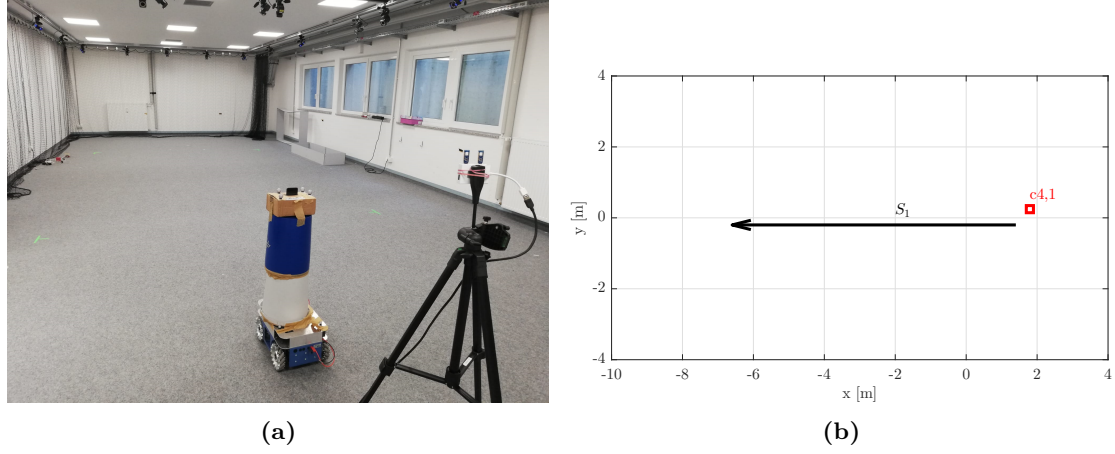


Figure 3.3: Indoor environment for measurement campaign 1 to acquire the path loss model.

(a) Indoor measurement setup with the robot and a BLE receiver $c_{4,1}$ mounted on a tripod for measurement campaign 1 to acquire the path loss model.

(b) Ground truth view of the setup with the path S_1 and the BLE receiver $c_{4,1}$.

The receiver $c_{4,1}$ receives the advertisement packets from the mobile device acting as BLE beacon, determines the RSSI and forwards the data over the serial port interface to a host machine. The ground truth position of the moving beacon and the static receiver is obtained by the optical motion capture system, which is used to determine the ground truth distance between the beacon and the receiver.

The path loss is determined by moving the robot with the mounted smartphone on it over a distance of 8 m in 10 cm steps. The RSSI of the three advertising channels is recorded in the standing phases of the robot. The recorded RSSI of the all three advertising channels 37, 38, and 39 plotted over the distance for the receiver $c_{4,1}$ are shown in Figure 3.4a. The RSSI of channel 37 are shown in green (—), channel 38 in purple (—) and channel 39 in orange (—). As depicted in Figure 3.4a, the advertising channels are subject to different path losses, showing fluctuations with varying degree in amplitude. For instance at a distance of 2 m, a fluctuation of 20 dBm is observed at channel 37, whereas fluctuations of only 5 dBm and 8 dBm are measured on the channels 38 and 39. Based on the first measurement campaign it is decided to average over all advertising channels as a countermeasure against multipath errors. The mean RSSI of all channels is depicted in light blue (—) in Figure 3.4a.

After the averaging over all advertisement channels shows a significant improvement in reducing the fluctuations in the RSSI, it is investigated, whether the extension of a second closely spaced BLE receiver can further reduce fluctuations, similar to the approach in [20]. The experiment is carried out again using two BLE receivers $c_{4,1}$ and $c_{4,2}$ spaced with a small distance from each other mounted in a USB hub, referred to as sniffer c_4 . The serial port data of the two dongles of the sniffer c_4 is recorded

simultaneously. Figure 3.4b shows the result of the experiment using two dongles as sniffer c_4 . The mean RSSI of the channels 37, 38, and 39 for receiver $c_{4,1}$ is shown in light blue (—) and for the second receiver $c_{4,2}$ in dark blue (—). The mean of both dongles is depicted in red (—). The mean of both

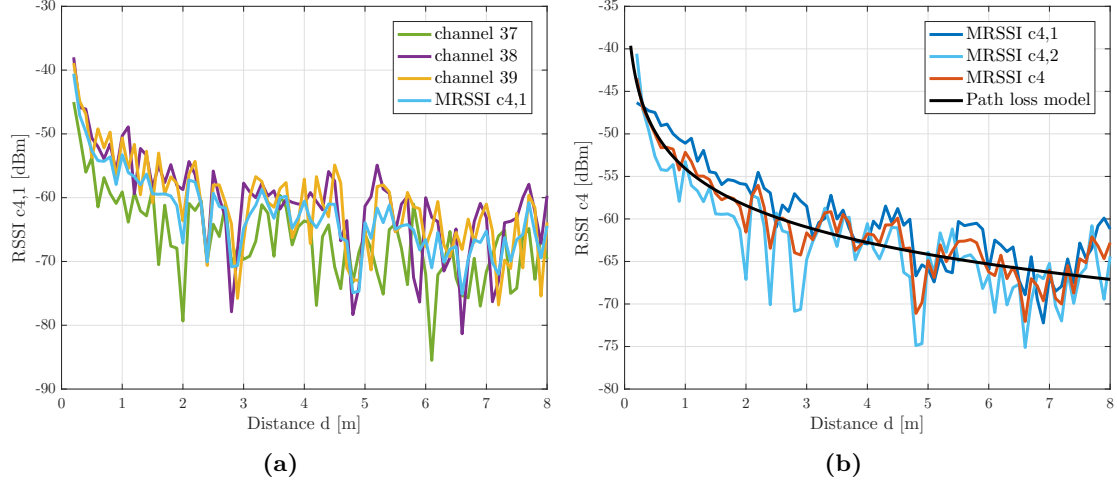


Figure 3.4: Indoor environment for measurement campaign 1 to acquire the path loss model.

(a) RSSI of the channels 37, 38, and 39 and mean mean received signal strength indicator (MRSSI) plotted over distance for BLE receiver $c_{4,1}$.

(b) RSSI of two closely spaced BLE receivers and combined RSSI of the sniffer c_4 and path loss model plotted over distance.

As can be seen in Figure 3.4b, both dongles obtain different multipath propagation and act as redundancy for each other. By using two dongles on sniffer c_4 , small scale fading might be reduced, the fluctuations caused by multipath propagation are more Gaussian and thus less biased and can be better utilized for distance and position estimation. With averaging over multiple channels and over multiple receivers a standard deviation of 6.1 dBm can be obtained, compared to a standard deviation of 8.2 dBm when only using channel 37.

With the recorded mean RSSI of sniffer c_4 and the known ground truth distance, the path loss model described in Section 2.1.3 can be determined with the OLS method. Therefore, the recorded RSSI is represented by the column vector $\mathbf{y} = [\text{RSSI}_1, \dots, \text{RSSI}_n]^T$ and the matrix $\mathbf{X}$ is set to

$$\mathbf{X} = \begin{pmatrix} 1 & \log_{10}(d_1) \\ \vdots & \vdots \\ n & \log_{10}(d_n) \end{pmatrix}. \quad (3.1)$$

The coefficients $\boldsymbol{\beta} = [\beta_0, \beta_1]^T$ representing the parameters of the path loss model in Equation (2.5) can be calculated with OLS by

$$\boldsymbol{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}. \quad (3.2)$$

The result of the OLS is depicted in Figure 3.4b where the mean RSSI of the two dongles is shown in red (—) and the OLS result giving the path loss model is shown in black.

3.4 Server-Based Distance Estimation

In this thesis, a server-based approach for a BLE-based IPS is considered, where mobile devices act as BLE beacons and static BLE devices act as BLE sniffers, that obtain RSSI measurements [11]. The design as a server-based IPS is not limited by the API restrictions of the smartphone manufacturers and is thus expected to be more flexible regarding the provided information necessary for the positioning task. The BLE sniffers are installed statically in the indoor environment and are connected, e.g. by wireless local area network (WLAN), to a server. The server is hosting a persistency layer and the positioning functionality. As described in Section 2.1.3 and showed with measurements in Section 3.3, the raw RSSI is subject to fluctuations caused by multipath propagation. To increase the distance estimation accuracy, averaging over multiple channels and multiple BLE receivers is identified as a suitable measure against fluctuations.

Therefore, each BLE sniffer c_i provides N_{c_i} BLE receivers $c_{i,j}$, closely distanced from one another. By computing the average RSSI over multiple BLE receivers, small scale fading is expected to be minimized and a better position estimation is possible, which is also shown in [17]. The term BLE receiver is used for each individual BLE receiver $c_{i,j}$ and BLE sniffer c_i for the device equipped with multiple BLE receivers. The mobile device B_b with $b \in \{1, \dots, N_b\}$ at position $\mathbf{r}_b(t)$ acting as a beacon regularly broadcasts BLE advertising packages. When a BLE package broadcasted from a mobile device B_b on channel $m \in \{37, 38, 39\}$ is received at the BLE receiver $c_{i,j}$ with $j \in 1, \dots, N_{c_i}$ of BLE sniffer c_i , $i \in 1, \dots, N_c$ at time t , the measured RSSI $\hat{z}_{B_b,i,j,m}(t)$ is persisted on the server [11].

The distance and position estimation algorithms hosted by the server are scheduled on a fixed sampled time grid t_k , with $T_g = t_k - t_{k-1}$ denoting the time between two adjacent time stamps. The distance between a BLE sniffer c_i and the mobile device B_b at position $\mathbf{r}_b(t_k)$ is estimated based on multiple RSSI measurements received at the BLE sniffer c_i . With this approach the distance estimation is not performed with every new arriving BLE measurement but with averaged measurements of all three advertisement channels of the two BLE receivers of the observed period T_g . This measure is supposed to massively reduce the required computational power and to also smoothen the RSSI data. To allow the time-grid-based averaging, the BLE sniffers and the server are time synchronized, e.g. using Network Time Protocol (NTP) [15]. To perform a RSSI-based distance estimation, the RSSIs are filtered by an averaging filter to obtain the MRSSI $\bar{z}_{B_b,i}(t_k)$. All RSSIs $\hat{z}_{B_b,i,j,m}(t)$ from the BLE sniffer c_i that are inside the time period $t_k - T_g < t$ are taken into account by the averaging filter. The length of the averaging period is represented by $N_{l,i}(t_k)$. As a result,

$$\mathbf{V}_{B_b,i} = \left[V_{B_b,i,1}, V_{B_b,i,2}, \dots, V_{B_b,i,N_{l,i}(t_k)} \right]^T \quad (3.3)$$

is a vector of length $N_{l,i}(t_k)$ that contains all RSSIs $\hat{z}_{B_b,i,j,m}(t)$ of BLE device B_b and BLE sniffer c_i , that are received during the time interval $t_k - T_g < t$. The MRSSI

$\bar{z}_{B_b,i}(t_k)$ is calculated as

$$\bar{z}_{B_b,i}(t_k) = \frac{1}{N_{l,i}(t_k)} \sum_{l=1}^{N_{l,i}(t_k)} V_{B_b,i,l} . \quad (3.4)$$

Building upon the first filter stage, a second stage acting as a moving average filter is proposed. For the moving average filter each MRSSI $\bar{z}_{B_b,i}(t_k)$ is stored in a vector $\mathbf{W}_{B_b,i}$ of length $M_{q,i}(t_k)$. The vector $\mathbf{W}_{B_b,i}$ then contains all MRSSI $\bar{z}_{B_b,i}(t_k)$ of the moving average window $t_k - M_{q,i}(t_k) \cdot T_g < t$. The moving average filtered MRSSI $z_{B_b,i}(t_k)$ is calculated as

$$z_{B_b,i}(t_k) = \frac{1}{M_{q,i}(t_k)} \sum_{q=1}^{M_{q,i}(t_k)} W_{B_b,i,q} . \quad (3.5)$$

This two staged filtering process is schematically depicted in Figure 3.5. The upper part shows the first stage establishing the fixed time grid t_k . The lower part shows the second stage implementing the moving average filter. The latest moving average value depicted with a bold frame is referred to as MRSSI measurement $z_{B_b,i}(t_k)$ of the sniffer c_i at time instant t_k which is further used estimate the distance to the mobile device B_b .

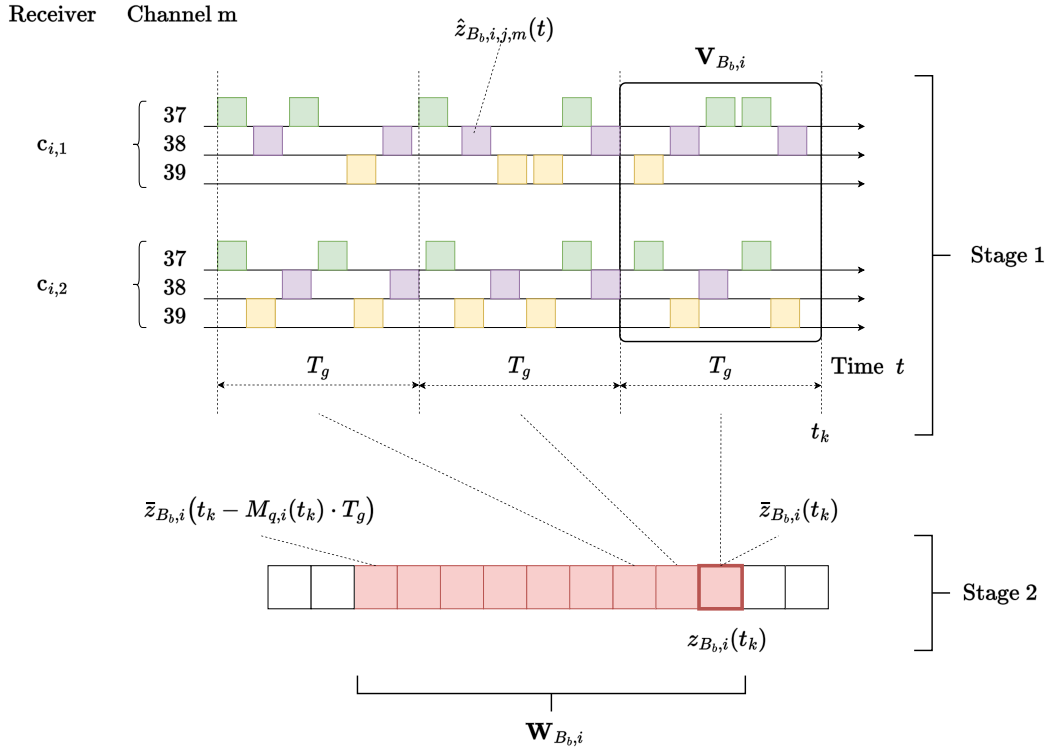


Figure 3.5: Two staged RSSI filtering of the raw RSSI to obtain less fluctuations for the distance estimation.

The distance $\hat{d}_{B_b,i}(t_k)$ between mobile device B_b and BLE sniffer c_i can finally be

estimated by solving Equation (2.3) for d ,

$$\hat{d}_{B_b,i}(t_k) = \sqrt{\frac{1}{\bar{P}_{B_b,i}(t_k)}} \sqrt{P_{B_b} G_{B_b} G_r} \cdot \left(\frac{\lambda}{4\pi} \right), \quad (3.6)$$

where $\hat{d}_{B_b,i}(t_k)$ denotes the estimated distance and $\bar{P}_{B_b,i}(t_k) = 1 \text{ mW} \cdot 10^{\frac{z_{B_b,i}(t_k)}{10}}$ is the MRSSI in linear domain. As mentioned before, the transmit power P_{B_b} of the mobile device B_b and antenna gains G_{B_b} and G_r are known or can be calibrated with the path loss model [11].

3.5 Transition Models

As described in Section 2.2.1, the tracking of the kinematic state of an object is often done by modeling the movement with a so-called transition model in combination with a Bayesian approach, in this case with a PF. Transition models exist for various number of different applications and thus have to be chosen according to the concrete scenario. This thesis aims to indoor locate a mobile device like a smartphone, which is often equipped with an inertial measurement unit (IMU). The following paragraphs thus focus on two different transition models, considering the cases without and with IMU data available.

3.5.1 Gaussian-Transition-Model

In the first scenario, the mobile device is assumed to not provide information about the orientation using an IMU. In this case, the state transition can be described using a Gaussian-Transition-Model [3], resulting in the following state representation of a particle $\mathbf{x}^{(j)}(t_k)$ as

$$\mathbf{x}^{(j)}(t_k) = \begin{pmatrix} p_x^{(j)}(t_k) \\ p_y^{(j)}(t_k) \\ v_x^{(j)}(t_k) \\ v_y^{(j)}(t_k) \end{pmatrix}, \quad (3.7)$$

with $p_x^{(j)}(t_k)$ and $p_y^{(j)}(t_k)$ describing the 2D position in x- and y-direction, $v_x^{(j)}(t_k)$ and $v_y^{(j)}(t_k)$ representing the 2D velocity vector in x- and y-direction in a 2D Cartesian coordinate system.

The particle state is then propagated from the previous step t_{k-1} to the current one t_k by using

$$\mathbf{x}^{(j)}(t_k) = \begin{pmatrix} 1 & 0 & T_g & 0 \\ 0 & 1 & 0 & T_g \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \mathbf{x}^{(j)}(t_{k-1}) + \begin{pmatrix} 0 \\ 0 \\ \sigma_{q_u}^2 \\ \sigma_{q_u}^2 \end{pmatrix}. \quad (3.8)$$

The elapsed time between the time instant t_k and t_{k-1} is expressed by T_g . Since a measurement of the velocity is not provided by mobile devices, the values $v_x^{(j)}(t_k)$ and $v_y^{(j)}(t_k)$ are sampled from a zero mean Gaussian process with standard deviation

σ_{q_u} to simulate the acceleration of the movement by a random process. The Gaussian-Transition-Model is schematically depicted in Figure 3.6.

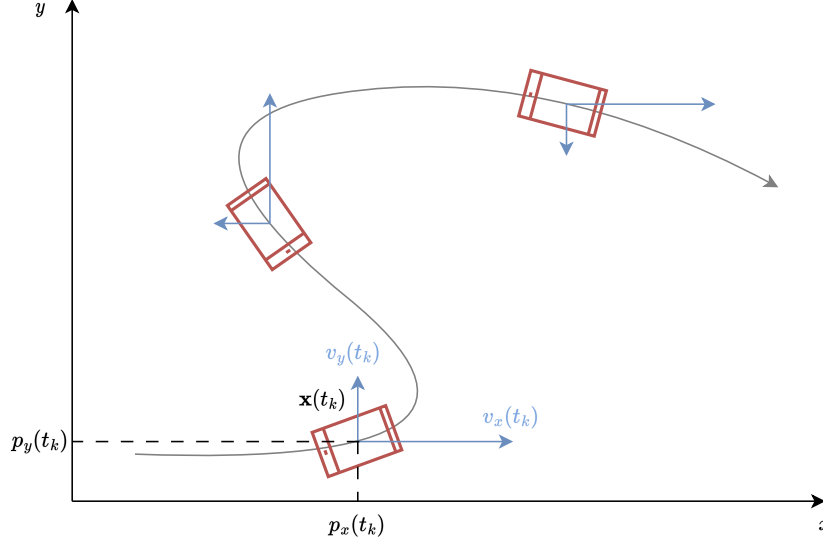


Figure 3.6: Illustration of the Gaussian-Transition-Model.

This model is for instance as well applicable for a robot equipped with omnidirectional wheels, meaning the robot can change its direction of movement without changing its orientation.

3.5.2 Inertial Measurement Unit (IMU)-Transition-Model

In the second considered scenario, the mobile device is equipped with an IMU. The IMU provides additional information about the three-dimensional (3D) acceleration and rotation velocity, which can be used in the state representation. The state of a particle $\mathbf{x}^{(j)}(t_k)$ is in that case represented by

$$\mathbf{x}^{(j)}(t_k) = \begin{pmatrix} p_x^{(j)}(t_k) \\ p_y^{(j)}(t_k) \\ v^{(j)}(t_k) \\ \varphi^{(j)}(t_k) \end{pmatrix}, \quad (3.9)$$

where $p_x^{(j)}(t_k)$ and $p_y^{(j)}(t_k)$ again describe the 2D position in x- and y-direction, $v^{(j)}(t_k)$ describes the magnitude of the velocity and $\varphi^{(j)}(t_k)$ describes the orientation, also referred to as heading or yaw angle. The heading angle can be estimated with the smartphone's internal IMU and is considered as the angle representing the smartphone's orientation projected on a horizontal plane when it is for instance being held in texting mode. The determination of the heading angle is focused on in Section 3.6.

Equation (3.9) can be extended by another state, the so-called yawrate $\Delta\varphi^{(j)}(t_k)$,

resulting in

$$\mathbf{x}^{(j)}(t_k) = \begin{pmatrix} p_x^{(j)}(t_k) \\ p_y^{(j)}(t_k) \\ v^{(j)}(t_k) \\ \varphi^{(j)}(t_k) \\ \Delta\varphi^{(j)}(t_k) \end{pmatrix}. \quad (3.10)$$

The yawrate $\Delta\varphi^{(j)}(t_k)$ describes the derivation of the heading angle $\varphi^{(j)}(t_k)$.

Using Equation (3.10) has the advantage of introducing more flexibility when it comes to the alignment of the world coordinate system and the local coordinate system of the smartphone. Normally, the coordinate systems of the smartphone and the world are not aligned, meaning the heading direction estimated by the smartphone does not represent the actual walking heading direction, since the heading angle of the smartphone is a measure relative to the initial heading. Equation (3.10) solves this problem by integrating the differential angle, i.e. the yawrate, with a given or assumed initial heading $\varphi^{(j)}(0)$.

In this case the particle state is propagated from the previous step t_{k-1} to the current one t_k by using

$$\mathbf{x}^{(j)}(t_k) = \mathbf{x}^{(j)}(t_{k-1}) + \begin{pmatrix} T_g \sin(\varphi^{(j)}(t_k)) \\ T_g \cos(\varphi^{(j)}(t_k)) \\ 0 \\ T_g \Delta\varphi^{(j)}(t_k) \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \sigma_{q_u}^2 \\ 0 \\ \sigma_{q_y}^2 \end{pmatrix}. \quad (3.11)$$

The IMU-Transition-Model builds upon a white noise model where inaccuracies are modeled using a zero mean Gaussian process with standard deviation σ_{q_u} for the magnitude of the velocity and σ_{q_y} for the yawrate.

Modeling the transition using the IMU-Transition-Model using Equation (3.10) is expected to lead to better results, since the heading angle and yawrate introduce another measurand to the state representation and the PF has more information about the current target state. The IMU-Transition-Model is schematically depicted in Figure 3.7.

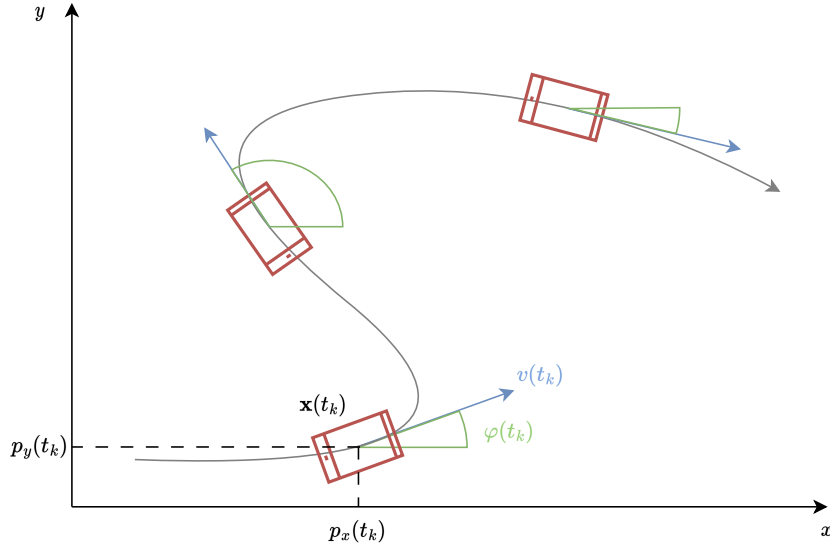


Figure 3.7: Illustration of the IMU-Transition-Model.

3.6 Determination of the Orientation of a Mobile Device

The modeling of the dynamics of an object to be tracked is often done by fusing sensor data from different kinds of sensors. As described in Section 3.5, for indoor tracking of a mobile device, further information about the movement is expected to lead to better accuracy. However, the variety of sensors suitable for indoor tracking on mobile devices is limited [2].

Typically, modern smartphones feature multiple optical sensors like multi-lens cameras, microphones, hygrometers, barometers, thermometers, light sensors, GNSS receivers for outdoor positioning, radio transceivers, and inertial sensors to determine the orientation. As already mentioned in Section 2.1, BLE is utilized for distance estimation. In the IMU-Transition-Model described in Section 3.5.2, the orientation is a suitable additional measurand. Orientation measurement is often realized by sensor fusion of inertial sensors like gyroscopes, accelerometers and magnetometers [14].

Gyroscopes measure angular velocities on the three axes of smartphones and can thus observe changes in the three rotational axes of the device (yaw, pitch and roll). Accelerometers record the total external specific forces applied to it, including accelerations and gravity. However, hardware sensors like gyroscopes or accelerometers are often subject to noise, drift, bias and deviation and thus cannot be directly utilized. To obtain angular measurands, gyroscopes are often integrated, whereby noise and bias often lead to drift. To use the result, different sensors that complement one another are typically fused [14].

If the right representation is adopted, it is simple to combine sensors that produce a 3D rotation. Different formats can be used to depict a rigid body's rotation in 3D space. For instance Euler angles represent a 3D rotation as a vector of three angles (yaw, pitch, roll), are easy to interpret, but suffer from gimbal lock. Another possible

representation is the 3×3 rotation matrix. Widely used are so-called quaternions, that use a real part and 3 imaginary parts similar to a complex number. Quaternions are elegantly handlable when it comes to further mathematical operations like 3D rotation interpolation [14].

For the purpose of sensor data fusion, different algorithms are available. The simplest version is a complementary filter. A complementary filter combines a high-pass filter for drift elimination of the integrated gyroscope and a low-pass filter for noisy accelerometer or magnetometer values. Often a drift-free angle can be obtained with good dynamic response.

A higher sophisticated but more computationally intensive fusion can be realized with a KF, that is based on the Bayesian Filtering, described in Section 2.2.1. KFs implement a feedback loop with a prediction and an update stage, assuming a linear system with Gaussian noise. An extension to non-linear systems is also possible.

A widely used fusion filter is the Mahony filter, which improves the complementary filter by applying a proportional integral controller to the error function between the orientation from the acceleration data and the orientation from the gyroscope data. The integral part of the controller is able to reliably remove a constant offset caused by biased acceleration or gyroscope data. This filter has two parameters to tune the results. The controller correction is also applied to the gyroscope data. The proportional error is calculated as the sum of the cross product between the gravity vector estimated by the orientation and updated by the gyroscope data and the gravity vector measured by the accelerometer [14]. In Section 3.6, the Mahony fusion algorithm is implemented to obtain the orientation of the smartphone.

3.7 Particle Filter-Based Position Estimation

To utilize BLE signals for positioning purposes, the localization problem can be expressed using a state-space representation. The state of the mobile device B_b at time instant t_k can be defined using the state-space description explained in Section 3.5. As in this case a dynamic system is considered, the state estimation can be described based on a discrete-time formulation using a transition and a measurement model [11].

The measurement model uses a potential non-linear function and measurement noise to associate the state vector to the measurements at time instant t_k . By using Equation (3.6), the distance

$$\hat{d}_{B_b,i}(t_k) = \|\mathbf{x}(t_k) - \mathbf{r}_{B_i}\| + n_m(t_k), \quad (3.12)$$

can be obtained where $n_m(t_k)$ describes the general measurement noise with standard deviation σ_m for the measurement.

From a Bayesian perspective, the transition model can be expressed by the conditional PDF $p(\mathbf{z}(t_k) | \mathbf{x}(t_k))$, which is referred to as transition prior distribution, where $\mathbf{z}(t_k)$ defines the set containing the MRSSI measurements at time instant t_k with

$$\mathbf{z}(t_k) = [z_1(t_k), z_2(t_k), \dots, z_{N_c}(t_k)]^T = [\hat{d}_{B_b,1}(t_k), \hat{d}_{B_b,2}(t_k), \dots, \hat{d}_{B_b,N_c}(t_k)]^T. \quad (3.13)$$

Here the availability of the MRSSIs from all N_c BLE sniffers at each time instant t_k is assumed whereby the adaption to a time-variant size of $\mathbf{z}(t_k)$ is possible as well.

In the Gaussian-Transition-Model, only distance estimates are used as measurements $\mathbf{z}(t_k)$ for the weight update in the SIRPF. Hence, the likelihood distribution $p(\mathbf{z}(t_k) | \mathbf{x}^{(j)}(t_k))$ of the measurement model can be expressed as

$$\begin{aligned} p(\mathbf{z}(t_k) | \mathbf{x}^{(j)}(t_k)) &= \prod_{i=1}^{N_c} p(\mathbf{z}^{(i)}(t_k) | \mathbf{x}^{(j)}(t_k)) \\ &= \prod_{i=1}^{N_c} \frac{1}{\sqrt{2\pi}\sigma_m} e^{-\frac{1}{2} \left(\frac{\|\mathbf{x}^{(j)*}(t_k) - \mathbf{r}_{B_i}\| - \hat{d}_{B_b, i}(t_k)}{\sigma_m} \right)^2}, \end{aligned} \quad (3.14)$$

where $\mathbf{x}^{(j)*}(t_k) = [p_x(t_k), p_y(t_k)]^T$ and independent MRSSI measurements and normally distributed measurement noise samples are assumed. For the distance measurement, $\sigma_m = 3$ m is empirically chosen. [11].

In the IMU-Transition-Model, two different kinds of measurements are considered for the weight update, namely distance estimates $\mathbf{z}(t_k)$ and yawrate measurements $\Delta\varphi_{B_b}(t_k)$. In a discrete-time system as a difference equation, the yawrate $\Delta\varphi_{B_b}(t_k)$ is calculated with

$$\Delta\varphi_{B_b}(t_k) = \varphi_{B_b}(t_k) - \varphi_{B_b}(t_{k-1}). \quad (3.15)$$

Hence, two distinct likelihood distributions are used to do the overall weight update. For the distance estimates $\mathbf{z}(t_k)$, the likelihood distribution described in Equation (3.16) is used. For the yawrate measurements, the likelihood distribution $p(\Delta\varphi_{B_b}(t_k) | \mathbf{x}^{(j)}(t_k))$ is formulated as

$$p(\Delta\varphi_{B_b}(t_k) | \mathbf{x}^{(j)}(t_k)) = \frac{1}{\sqrt{2\pi}\sigma_{\Delta\varphi}} e^{-\frac{1}{2} \left(\frac{\Delta\varphi^{(j)}(t_k) - \Delta\varphi_{B_b}(t_k)}{\sigma_{\Delta\varphi}} \right)^2}. \quad (3.16)$$

For the likelihood distribution of yawrate of the IMU-Transition-Model, $\sigma_{\Delta\varphi}$ is set to 0.5 rad/s.

The overall weight update is therefore relying on two different likelihood distributions and is performed by combining the two likelihood distributions with

$$p(\mathbf{z}(t_k) | \mathbf{x}^{(j)}(t_k)) \cdot p(\Delta\varphi_{B_b}(t_k) | \mathbf{x}^{(j)}(t_k)). \quad (3.17)$$

3.8 Further Measures to Enhance the Position Estimation

As investigated in [11], the quality of the position estimation is heavily influenced by the accuracy of the distance estimations between the mobile device B_b and the static BLE sniffers c_i . Furthermore, the spatial setup of the sniffers has an impact on the positioning performance, where according to [11] a setup with fewer sniffers does not necessarily lead to a worse positioning performance. Building on these two insights, two more methods concerning the improvement of the positioning in case of severe fluctuations are proposed.

Choosing the Sniffers with the Highest RSSI

One possible countermeasure to severe fluctuations can be the elimination of sniffers with low RSSI measurements and thus high distance estimates. Following this approach, only the N_c^* sniffers with the highest RSSI measurement are chosen for the position estimation, with $N_c^* = \alpha \cdot N_c$. In this case, the implementation has to deal with a time-variant size of $\mathbf{z}(t_k)$. To avoid ambiguities in the position estimation, it is suggested to choose α , that $N_c^* \geq 3$ is fulfilled.

One problem with this approach is the possible concentration of the resulting N_c^* sniffers in one point, leading to a poor positioning performance. To avoid the concentration in one point, a second condition concerning the all-to-all distances between the sniffers could be introduced when selecting the relevant sniffers. This approach is expected to lead to better estimations in bigger environments with a high number of sniffers.

Dynamic Weighting in the SIRPF

As described in Section 2.2.2, the contribution of a particle $\mathbf{x}^{(j)}(t_k)$ to posterior distribution $p(\mathbf{x}(t_k) \mid \mathbf{z}(t_{1:k}))$ is determined by its weight $w^{(j)}(t_k)$ which is derived from the measurement $\mathbf{z}(t_k)$ using Equation (3.14) with a fixed specified standard deviation. By introducing a distance-dependent standard deviation $\sigma_m = f(\hat{d}_{B_b,i}(t_k))$, particles with great distances can be assigned a lower weight, resulting in a smaller contribution to the position estimation, where $f(\cdot)$ denotes a possible non-linear function to determine the standard deviation σ_m . This procedure is referred to as dynamic weighting. Similar to activation functions in artificial neural networks, a parameterized sigmoid function is used, denoted as

$$\sigma_m = f(\hat{d}_{B_b,i}(t_k)) = a + b \cdot \frac{1}{1 + e^{-c \cdot \hat{d}_{B_b,i}(t_k) + d}}, \quad (3.18)$$

with the parameters a , b , c , and d . The parameters are empirically determined by positioning runs and depend on the concrete scenario. The likelihood distribution $p(\mathbf{z}(t_k) \mid \mathbf{x}^{(j)}(t_k))$ can then be expressed as

$$\begin{aligned} p(\mathbf{z}(t_k) \mid \mathbf{x}^{(j)}(t_k)) &= \prod_{i=1}^{N_c} p(\mathbf{z}^{(i)}(t_k) \mid \mathbf{x}^{(j)}(t_k)) \\ &= \prod_{i=1}^{N_c} \frac{1}{\sqrt{2\pi} f(\hat{d}_{B_b,i}(t_k))} e^{-\frac{1}{2} \left(\frac{\|\mathbf{x}^{(j)*}(t_k) - \mathbf{r}_{B_i}\| - \hat{d}_{B_b,i}(t_k)}{f(\hat{d}_{B_b,i}(t_k))} \right)^2}. \end{aligned} \quad (3.19)$$

Chapter 4

Implementation of the Server-Based Indoor Positioning System

This chapter focuses on the concrete implementation of the proposed concept of Chapter 3. The first part copes with the implementation a Matlab version. Thereafter, the embedded firmware of the nRF52840 Dongle is focused on. After that, the implementation of the Android application is presented. In the end, the server-based approach is implemented as an actual embedded IPS in C++ that supports live positioning.

4.1 Matlab Version

The proposed concept of the server-based BLE IPS is first implemented in Matlab. The aim is to investigate whether accurate positioning is possible using the selected hardware components and which initial parameters and tuning values are required for the proposed algorithms. The Matlab version follows a post-processing approach, where the BLE data is first recorded and stored and the algorithms are evaluated and tested afterwards. It can for instance be used for calibration of the path loss model of each individual sniffer. In this case the BLE data is first recorded by the embedded IPS in a log file. The data can then be read in and be processed using Matlab. That way, the path loss model coefficients are determined, which can then be fed back to the embedded IPS. Also, the Matlab version can be utilized to test the proposed algorithms and to compare the results of the post-processed data with the live processed data from the embedded IPS. The logged data file from the server is read and the data of the format described in Equation 4.2 is parsed into a nested database-like structure. The data is grouped by sniffer identification number (ID), by BLE receiver ID and by advertising channel with associated timestamps. With this data structure, a detailed analysis of every single channel of every BLE receiver of every sniffer is possible and the necessary filtering algorithms can be developed and implemented according to the requirements.

4.2 nRF52840 Dongle Firmware

As mentioned, the IPS in this thesis is implemented relying on a server-based approach, where the mobile device, e.g. a smartphone, acts as a transmitter. The distance estima-

tion is done using multiple static sniffers connected to a server hosting the positioning algorithm. The first step to establish the server-based approach is the implementation of a BLE receiver on the nRF52840 Dongle. The nRF52840 Dongle is chosen because of the provided access to the BLE radio. It is possible to select the scanning channel, scanning window, and scanning interval at your discretion since the firmware is developed in embedded C. The program can be split into three parts: Initialization, BLE radio management, and the receiving callback.

In the first step, the nRF52840 Dongle is configured by initializing the board hardware, the USB interface, a 1 ms timer, the BLE radio, and by setting a receiving callback for received BLE advertisements. Next, the reading of the dongle-unique serial number is used to assign a unique ID to the dongle based on a table lookup. The program then runs into a while loop, which is responsible for managing the advertising channel, scan window d_s and scan interval T_s .

Based on the previously initialized 1 ms timer, the checks for the scan interval T_s and scan window d_s are performed. When the timer exceeds the threshold for the scan window d_s , the BLE radio is disabled and the time is remembered. When the timer exceeds the threshold for the scan interval T_s , the BLE radio is prepared with the currently selected scanning channel and then activated and put into receiving mode. The time is remembered. The next channel is prepared for the next scan interval T_s . This mechanism ensures a rolling scanning of the advertising channels.

When a BLE packet is received by the dongle, the previously set callback is asynchronously triggered. The callback checks the CRC and parses the received PDU in a predefined structure in case of a valid CRC. The nRF52840 Dongle receives all BLE packets of surrounding BLE devices. To distinguish between a device of interest and other BLE devices of the surroundings, the advertising data is analyzed. The device of interest is the Android smartphone Google Pixel 5 used in the experiments. On the Android side, the advertising data is set to a specific value, which also known by the nRF52840 Dongle. Only incoming BLE packets with this value are accepted and used for further processing. In some cases, also the advertising address in the BLE packets could be utilized to identify a BLE device. On Android devices, the BLE advertising address is periodically changing as a security measure and thus cannot be used. The Android side is focused in Section 4.3. If the BLE device is identified as a device of interest, a predefined data structure is reported over the serial port with USB. The data includes the receiver ID, the obtained advertising address and data, the currently scanned advertising channel, the obtained RSSI value from the BLE radio API, and the current timer counter value. An example of the data reported by the dongle over USB with the structure elements is given by

$$\begin{array}{ccccccc}
 & \text{Advertising address} & & \text{Heading} & \text{RSSI} & & \\
 \underbrace{6; 69:d9:e2:a4:b7:ae; Pixel 5; 17; 2; 37; 59; 89494} & & & & & & \\
 \text{Receiver ID} & \text{Device name} & \text{Channel} & \text{Counter} & & & \\
 & \text{Advertising data} & & & & &
 \end{array} \quad (4.1)$$

This data can be used at a receiving serial port host for further processing. The mentioned procedures are depicted in the flowchart in Figure 4.1.

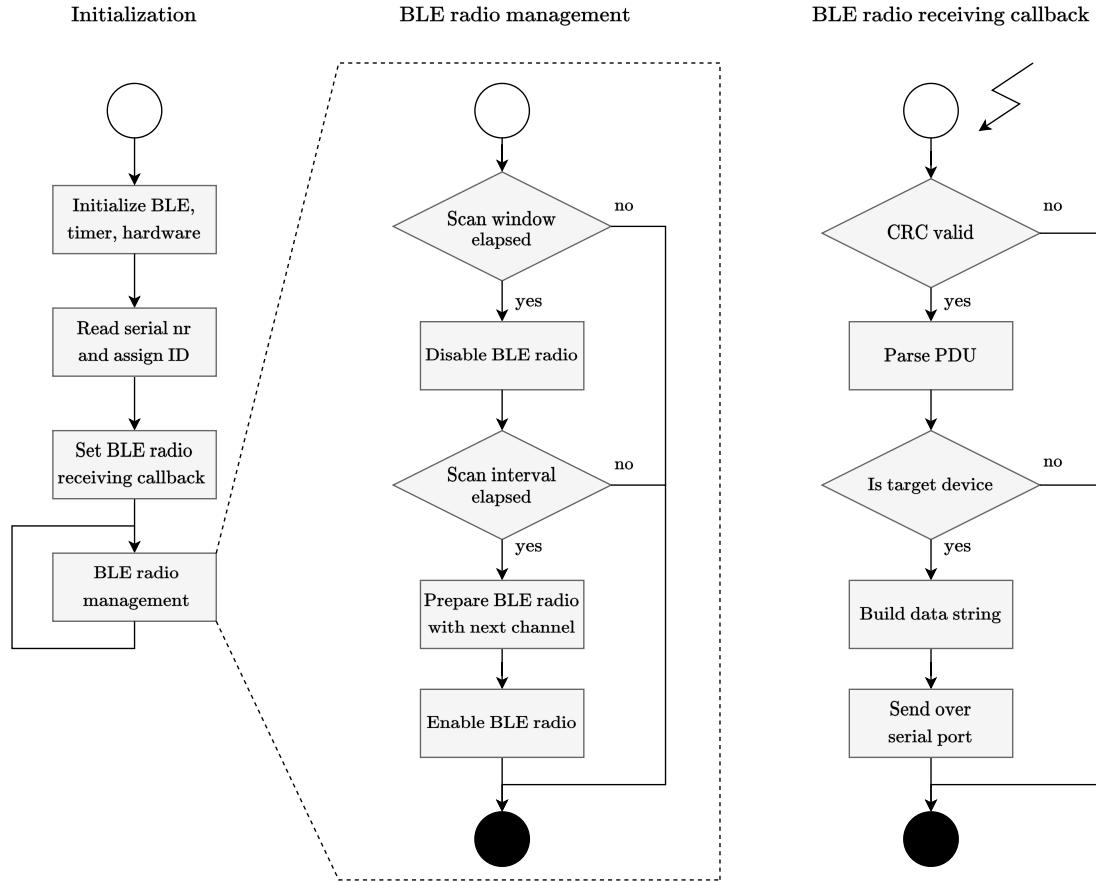


Figure 4.1: Flowchart of the nRF52840 Dongle firmware implementation showing the three main parts: Initialization, BLE radio management, and BLE radio receiving callback.

4.3 Android Application

In the server-based approach, the smartphone acts as a BLE transmitter, also referred to as BLE beacon. Considering orientation measurements for the position estimation is expected to lead to better positioning performance. The determination of the orientation is established by sensor fusion of the inertial sensors of the smartphone. The implementation of the BLE beacon and the orientation determination is combined in a single Android application, which is focused on in the following sections.

4.3.1 BLE Beacon

The Android platform provides Bluetooth stack support, which allows a device to wirelessly communicate data with other Bluetooth devices through the Android Bluetooth APIs. The BLE API is supported since Android 4.3 (API 18) [21]. As mentioned in Section 2.1.2, for distance and further position estimation purposes, a BLE advertiser

with undirected, unconnectable, and unscannable advertising events is sufficient. For the smartphone to work as a BLE beacon, the minimum required API level is set to API 26, which enables the application to dynamically modify the BLE advertising data at runtime. That is especially useful when transmitting BLE advertising data which is not constant. As an advertiser, the Android class `BluetoothLeAdvertiser` is used. Settings regarding the advertising parameters can be made with the Android class `AdvertisingSetParameters`. The advertisement is set to not connectable, the advertising interval T_a is set to `INTERVAL_MIN` to ensure the advertising events happen as often as possible. According to [10], the advertising interval T_a is therefore set to 100 ms. Moreover, the BLE radio transmitter power is set to `TX_POWER_MAX`. As described in Equation (2.3), the received power depends on the transmitter power. It is expected to acquire a more reliable RSSI with a transmission power as high as possible. The data being advertised is configured to contain the device name and custom data, which can be added as so-called service data. Therefore the Android class `ParcelUuid` is utilized to generate a 16 byte Universally Unique Identifier (UUID) [21]. The custom data contains the current heading angle $\varphi_{B_b}(t)$, which is described in Section 3.6.

4.3.2 Integration of the Inertial Data Fusion

The implementation of the proposed sensor data fusion in the Android application is realized by using the Android system service `SENSOR_SERVICE` in combination with the Android class `SensorManager`. The `SensorManager` is utilized to access the hardware sensors `TYPE_ACCELEROMETER`, `TYPE_GYROSCOPE`, `TYPE_MAGNETIC_FIELD` of the Android class `Sensor`. The `SensorManager` is then used to register a `SensorEventListener` to each sensor with a sampling time of `SENSOR_DELAY_FASTEST`. The `SensorEventListener` is responsible for the triaxial data acquisition of every sensor [21].

The next step involves the periodical execution of the fusion filter, in this particular case, a Mahony Filter is implemented according to [14]. The timer of the class `Timer` is scheduled with a sampling rate of 100 Hz and an initial delay of 1 s. The fusion is then performed in a thread by passing the sensor values from the gyroscope, accelerometer, and magnetometer to the Mahony fusion algorithm to perform an update with the new values. The fusion result can be obtained in the format of Euler angles, whereby also quaternion is possible. Internally, the Mahony algorithm builds upon quaternions.

As the acquisition of the orientation of the smartphone is implemented, the heading angle has to be forwarded to the positioning algorithm running on the server. This can be done by using a separate data link, e.g. a Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) connection. Since the positioning on the server fuses the orientation with the same sampling rate T_g as the distance measurements, it is sufficient to embed it as advertising data in the BLE advertisement packet. The heading angle can be processed by the sniffer and server. Before Android API version 26, a dynamically at runtime changeable BLE advertising data was not supported but the advertising has to be restarted [21]. In API 26, the Android class `AdvertisingSet` was introduced, which is utilized in this thesis. When the fusion is performed, the obtained heading angle $\varphi_{B_b}(t)$ is extracted from the fusion result and is included in the advertising data using the Android function `setAdvertisingData` after the device name as described in Section 4.3.1.

For developing purposes, the sensor and fusion values are also displayed using text fields. The advertising can be started using a button after the Bluetooth functionality is enabled in the system settings. A screenshot of the application is depicted in Figure 4.2.

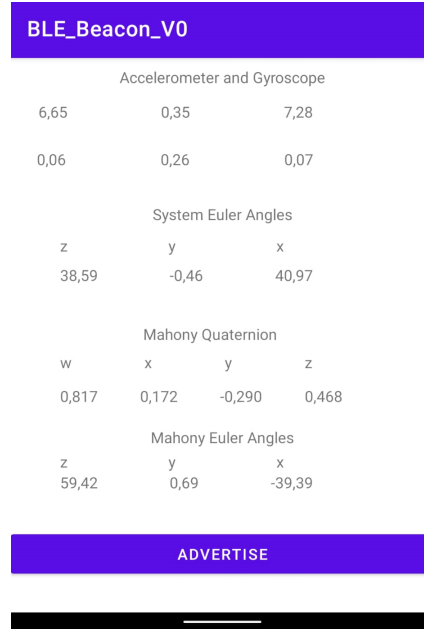


Figure 4.2: Screenshot of the Android application displaying the sensor values of the inertial sensors, the system Euler angles, and the result of the implemented Mahony fusion algorithm. The button at the bottom is used to start the BLE advertising in case the Bluetooth functionality is enabled.

4.4 Embedded Implementation

Based on the developed concept described in Chapter 3, an embedded implementation that supports live positioning is realized. First, an embedded platform has to be chosen. After that, the proposed server-based architecture is realized and the algorithms are implemented.

The embedded IPS is supposed to be flexible for modifications and future extensions as well as scalable, is supposed to cope with big, multiple-room environments, and may as well support future multiple target tracking with live positioning functionality. According to the required features, a powerful host platform is demanded. The server-based architecture is decided to be based on the network protocols TCP or UDP. The network communication is responsible for the data exchange between the sniffers and the server.

As mentioned in Section 3.2, a nRF52840 Dongle is used as a BLE receiver. For communication with a host platform, serial port support is required to read the data from the dongle and further process it. The dongle itself does not provide a time base, this might also be done by the host platform. To perform a time-grid-based averaging

mentioned in Section 3.4 a shared time base has to be established among all participants in the server-based architecture, meaning a time synchronization is necessary.

To meet performance requirements, not rely on interpreted or intermediate-language-based programming languages, access low-level hardware like serial ports, and allow a manageable programming effort, C++ is chosen as a target embedded programming language.

Regarding the aspects mentioned in the above paragraphs, it is decided to build the embedded implementation upon already existing and in practice established hardware. As a host platform for the sniffers and the server, the Raspberry Pi 4 platform is deployed. It provides a 64 bit Quad core Cortex-A72 (ARM v8) and up to 8 GB RAM [13]. As an operating system, the Linux-based Raspberry Pi OS Buster is utilized.

4.4.1 Server-Based Architecture

With the Raspberry Pi 4 based setup, it is possible to establish the proposed server-based architecture. The IPS consists of multiple static sniffers, whereby a sniffer is a system consisting of a Raspberry Pi 4 and two nRF52840 Dongles acting as BLE receivers. The two dongles act as redundancy helping each other to compensate for small-scale fading resulting from multipath propagation. The dongles receive BLE advertisements of a target of interest to be tracked and report the received data over the serial ports. The Raspberry Pi 4 acts as a local processing unit and reads the serial data from the dongles and processes it. Since the positioning task is done by the server, the sniffers have to forward the processed data to the server. As in this particular case the data from the sniffers is streaming data, the network protocol UDP is sufficient to send the data from the sniffers to the server, where the communication between the sniffers and the server is planned to be unidirectional.

The server receives the streaming data from the sniffers over UDP. The data is then parsed, locally persisted, and pre-processed in terms of filtering described in Section 3.4. After the pre-processing, the server performs the positioning with a SIRPF using the filtered data described in Section 2.2.4. The server is executed simultaneously on one Raspberry Pi 4 that also runs a sniffer, both running in different processes.

As mentioned in the above paragraphs, the server and the sniffers have to be time synchronized to perform the time-grid-based averaging. Time synchronization for network devices is often realized using NTP, which calculates time differences and creates a feedback loop by adjusting the clock frequency accordingly and is thus capable of achieving millisecond accuracy [15]. For the positioning server, a NTP server from an internet server pool is used. At the same time, the positioning server can act as a local NTP server for the sniffers. With this approach, only the positioning server needs an internet connection since the sniffers only rely on the local NTP server of the positioning server in the local network. The proposed server-based architecture including all data flows is depicted in Figure 4.3.

4.4.2 Sniffer Implementation

As mentioned in Section 4.4.1, the sniffers are responsible for reading data from the connected dongles and forwarding it to the server. The sniffer main program is implemented in C++ without external libraries. For the hardware access like serial port and

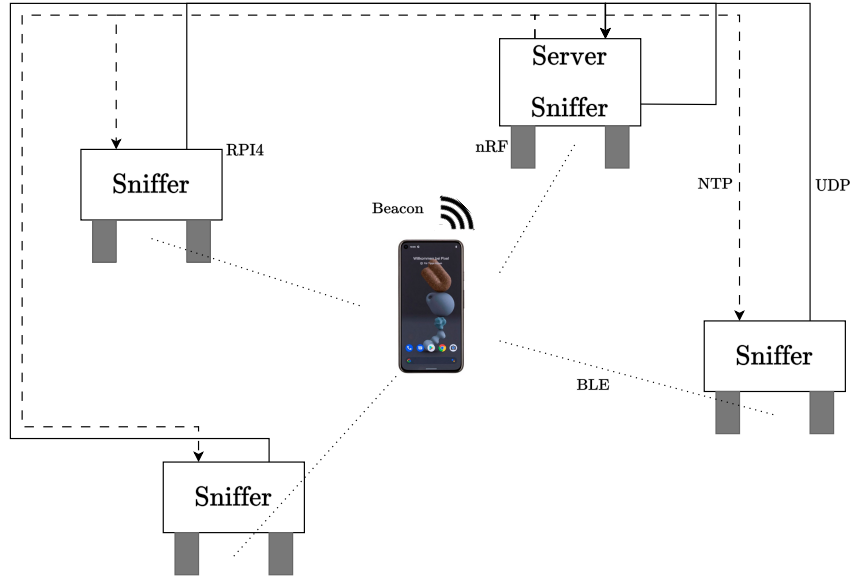


Figure 4.3: Server-based system architecture: The sniffers equipped with two BLE receivers receive the BLE packets from the advertising mobile device and forward the data to the server with UDP. The server does the positioning and synchronizes the sniffers using NTP.

socket communication, C libraries are utilized.

To send the data to the server using UDP, the Internet Protocol (IP) address of the server is to be known. The used local network equipment provides a Dynamic Host Configuration Protocol (DHCP) server that automatically assigns IP address for all participants. The server's IP address is thus determined with name resolution. To perform a name resolution, the network name of the server has to be set in advance. The IP address of the server can as well be set as static and would thus be a constant value for the sniffer. In this case, a DHCP server on the network side and a name resolution would not be necessary, resulting in less flexibility in case of network changes.

After the name resolution, two serial port communication interfaces are requested to open with `ioctl` system calls. After that, the two serial port interfaces are alternately polled for available data to be read in a loop. When available serial data is read, a Coordinated Universal Time (UTC) timestamp, the time difference to the previous data, and the sniffer ID are prepended. This data frame is sent over the UDP socket to the server. The full data frame that is sent to the server is composed like

$$\underbrace{2023-06-22\ 14:08:38.889}_{\text{UTC timestamp}}; \underbrace{2558.120; 3}_{\substack{t \\ dt}}; \underbrace{c1}_{\substack{\text{Sniffer ID} \\ \text{Receiver ID}}}; \underbrace{6; 69:d9:e2:a4:b7:ae}_{\text{Advertising address}}; \underbrace{\text{Pixel 5}}_{\substack{\text{Device name} \\ \text{Advertising data}}}; \underbrace{17; 2; 37}_{\substack{\text{Heading} \\ \text{Channel}}}; \underbrace{59; 89494}_{\substack{\text{RSSI} \\ \text{Counter}}}, \quad (4.2)$$

where the sniffer ID represented by the value `c1` is utilized to distinguish between the sniffers on the server.

4.4.3 Server Implementation

As indicated in Section 4.4.1, the server acts as the central part that performs the necessary steps to do the indoor positioning. In the next sections, the C++ server implementation is described.

Sniffers Ground Truth Position and Individual Path Loss Model

To perform the indoor positioning, the server must be provided with knowledge about the ground truth positions of the sniffers and the path loss model of each sniffer of the current setup. In this case, both parameters are determined with the optical motion capture system mentioned in Section 5.2 and are stored individually for each sniffer in a class `Anchor`. The information of all sniffers is organized in a constant single-key-single-value structure `std::unordered_map<std::string, Anchor>` where the sniffer ID is utilized for identification. The sniffer IDs to be used for the positioning can be specified, enabling the testability with different sniffer setups.

UDP Server Socket

The server receives the data in the format described in Section 4.4.2 from the sniffer over UDP. To do so, the server creates a UDP server socket on its local IP address and a specified port. This socket is used for blocked receiving of the UDP data from the sniffers in a while loop. The blocked receiving is sufficient since there is no need for the server to perform other calculations when there is no available data from the sniffers. When UDP data is received, it is parsed into an object of the class `RxData`, which encapsulates all properties listed in Section 4.4.2.

UDP parsing

If the data is parsed successfully, the parsed object is inserted into a single-key-multiple-value structure `std::unordered_multimap<std::string, RxData>` where the sniffer ID is again utilized for identification. This structure records all data for the time window T_g and is cleared when a new window starts. A `std::unordered_multimap` supports multiple values for one key in contrast to a `std::unordered_map`. With this structure, it is possible to persist an incoming parsed object to the associated sniffer ID. This reduces the effort of mapping and searching in the further processing steps.

Initial Heading and Heading Alignment

In case of the first received UDP packet, the heading $\varphi_{B_b}(t_k)$ is extracted from the packet and set as the initial heading $\varphi_{B_b}(0)$. By subtracting the initial heading $\varphi_{B_b}(0)$ from all following angles, the offset is corrected and the resulting initial heading is 0° . To align the estimated heading of the SIRPF with the actual walking direction, the initial walking direction $\varphi_W(0)$ is added. As a result for the IMU-Transition-Model, every particle state $\varphi^{(j)}(t_k)$ is initialized using

$$\varphi^{(j)}(t_k) = -\varphi_{B_b}(0) + \varphi_W(0) . \quad (4.3)$$

The initial walking direction $\varphi_W(0)$ is assumed to be known for the concrete scenario.

Time-Grid-Based Averaging

In the while loop, the program then compares the UTC timestamp extracted from the most recently received UDP packet to the one of the previous time window T_g . If the sampling period T_g of the time grid is exceeded, a SIRPF iteration is triggered. The server does not use the UTC timestamp when the UDP packet is received at the server, but uses the UTC timestamp, the serial data was read at the sniffer which is embedded in the UDP packet. With this approach, the problem of possibly different UDP network traveling times of the UDP packets is avoided.

Available Sniffers

Before the SIRPF can perform a position estimation, several pre-processing steps have to be run through. First, the available sniffers of the recorded time window are extracted. With this approach, it is not necessary for all sniffers to be available for the calculation all the time. This is for instance the case due to too great a distance to a specific sniffer. The available sniffers are then filtered to finally get the sniffers that should be used and also are available.

RSSI Pre-processing

After that, the MRSSI of the recorded time window is calculated. This procedure implements the first filter stage of the RSSI pre-processing explained in 3.4. The MRSSI is then persisted to enable the moving average filter of a desired length. This realizes the second filter stage explained in Section 3.4.

Distance Estimation and SIRPF Positioning

With the filtered MRSSI and the path loss model of the anchors to be used, the distances between the target to be tracked and the anchors to be used can be determined based on the filtered MRSSI. In case of the first iteration, the SIRPF is initialized, or else the prediction is performed. Afterwards, the current heading $\varphi_{B_b}(t_k)$ of the most recent UDP packet is read, the heading alignment is performed and the differential to the previous iteration, also referred to as yawrate $\Delta\varphi_{B_b}(t_k)$, is calculated. With the estimated distances between the target and the anchors to be used, with the ground truth positions of the anchors to be used, and with the yawrate $\Delta\varphi_{B_b}(t_k)$, the data is now ready for the SIRPF weight update. After the update, the MMSE from the particle set is calculated to get a point estimate. Finally, the particles are resampled and the time window T_g is cleared for the next iteration.

Data Logging

To also allow post-processing of the recorded data, all UDP packets from the sniffers are stored in a log file at the moment they are received. To evaluate the positioning results from the server, the currently available MRSSI, distance estimates, and the states of the MMSE of the particle set are logged in a separate log file.

The structure of the server program is depicted in Figure 4.4 as class diagram. The procedure of the main program of the server is shown in Algorithm 4.1.

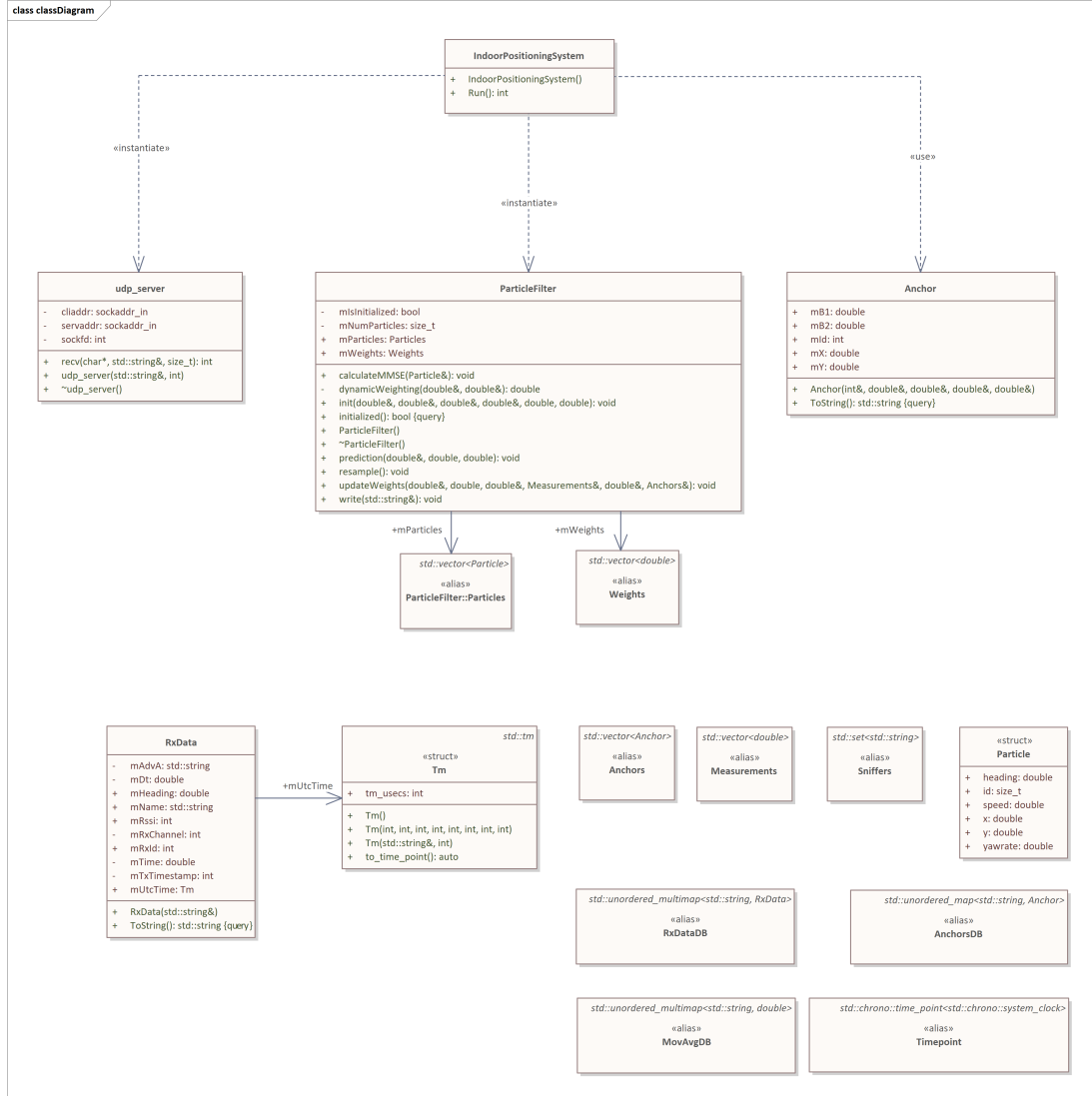


Figure 4.4: Class diagram of the server main program.

4.5 Offline and Live Positioning Mode

The C++ server program is designed to work as a hardware-in-the-loop (HIL) system to validate the functionality of the implementation with already available log data as well as with live data from the sniffers. The switch between HIL and live mode can be done with a single C++ macro. In HIL mode, the server reads an already recorded log file line by line, while in live mode the data is acquired from the UDP connections to the sniffers. It is possible to provide the HIL mode with the same data as the Matlab model.

Algorithm 4.1: Server main program

```

1: Define all sniffers of the setup
2: Define sniffers to be used
3: Open UDP server socket
4: while 1 do
5:   Read UDP data from the socket, log in file and parse in object
6:   Insert object in time window
7:   if FirstUDP then
8:     Set initial heading from first UDP packet
9:   if Time window  $T_g$  elapsed since previous iteration then
10:    Extract available sniffers from time window  $T_g$ 
11:    Filter sniffers with sniffers to be used
12:    Extract MRSSI measurements of time window  $T_g$ 
13:    Compute moving average with MRSSI
14:    Estimate distance with individual path loss model between target and sniffers
    to be used
15:    if FirstRun then
16:      Initialize SIRPF
17:    else
18:      SIRPF prediction
19:      Heading alignment and yawrate computation
20:      SIRPF weight update
21:      SIRPF calculate MMSE
22:      Log MMSE in file
23:      SIRPF resample
24:      Clear time window  $T_g$ 

```

With the HIL mode is thus possible, to perform offline tests, where no connection to the sniffers is necessary. After the result data from the SIRPF of the HIL mode is logged, it can be compared with the Matlab results. Due to stochastic processes in the SIRPF and different random seeds, the results slightly differ.

Chapter 5

Evaluation of the Server-Based Indoor Positioning System

The implementation described in Chapter 4 is evaluated in this chapter by conducting multiple measurement campaigns and comparing the results of the described algorithms.

5.1 Testbed and Experimental Setup for Data Acquisition

A second measurement campaign is conducted at the location named in Section 3.3. This time, the indoor environment is equipped with $N_c = 6$ static BLE sniffers mounted on tripods at a height of 1.4 m above the ground, depicted in Figure 5.1a.

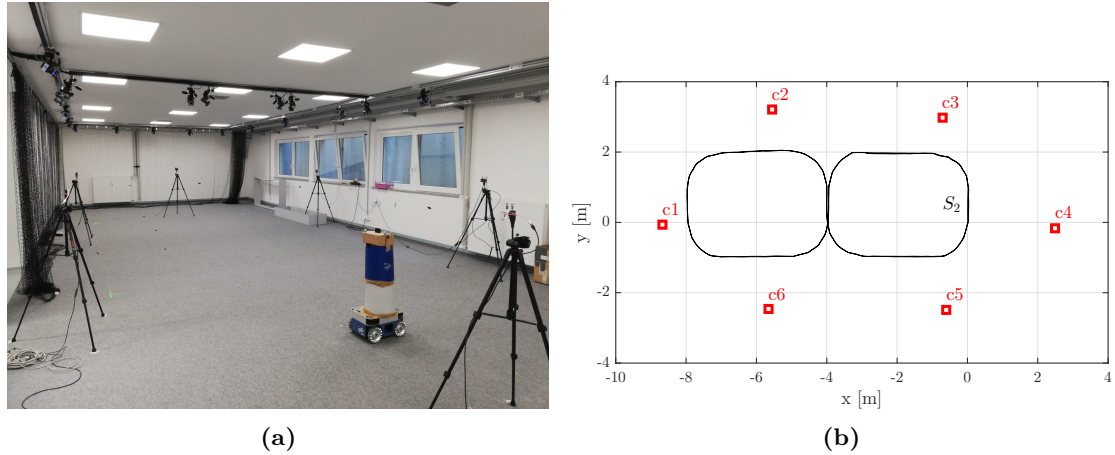


Figure 5.1: Indoor environment for measurement campaign 2.

- (a) Indoor measurement setup with the robot and the BLE sniffers c_i mounted on tripods for measurement campaign 2 to evaluate the distance and position estimation algorithms.
(b) Ground truth view of the indoor environment with the traversed path S_2 for the moving robot and the sniffers c_i .

The sniffers c_i , located at $\mathbf{r}_{c_i}$ with $i \in \{1, \dots, 6\}$, are units equipped with two

closely spaced nRF52840 Dongles acting as BLE receivers mounted in a USB hub. The 3D ground truth positions of the sniffers on the tripods are obtained using the optical motion capture system.

The movement of the mobile device is performed by a controllable, omnidirectional-wheeled mobile robot. The robot can be configured with a predefined trajectory where it uses positioning data from the optical motion capture system to autonomously traverse the path. In Figure 5.1b the indoor environment is illustrated as a view from the ground truth system. The traversed path S_2 is shown in black, and the placed static tripods with the sniffers c_i are depicted in red. The robot is equipped with a smartphone on top, which is acting as a BLE beacon broadcasting BLE advertisement packets.

The BLE advertisement packets are detected by the nRF52840 Dongles, read by the serial port of the sniffers and reported to the server, where the data is logged with associated UTC timestamps for later post-processing. The optical motion capture system simultaneously records the 3D ground truth position of the moving robot. The ground truth dataset is available as a text file and consists of the UTC timestamp, the 3D position, and the 3D orientation represented as a quaternion.

5.2 Preparation of the Recorded BLE and Ground Truth Data

After the BLE data and the ground truth data are recorded, both datasets need to be prepared for further processing and determining the path loss model. First, both datasets are read, and the UTC timestamps are compared. The sample rate of the ground truth dataset is 1 kHz and thus higher than the advertising rate of the smartphone. Hence, only the relevant data is extracted from the ground truth dataset by using a nearest neighbor search on the UTC timestamps of both sets. The first UTC timestamp is subtracted from all subsequent timestamps as an offset to start at time $t_k = 0$ and the dataset is converted to seconds domain to later establish a time-grid for filtering and positioning. Afterward, the BLE data is processed and organized as described in Section 4.1.

5.2.1 Individual Path Loss Model Calibration

After the data is prepared for each sniffer, further processing can be conducted. Due to the knowledge of the ground truth position from the optical motion capture system and now available RSSI data from the sniffers, an individual calibration of the path loss model described in Section 3.3 is possible. In comparison to measurement campaign 1, more data with a higher time resolution is available. Since the robot is not driving a straight line anymore but a trajectory, the RSSI data is expected to look more Gaussian. In reality, the RSSI is expected to still be biased and non-Gaussian due to multipath. However, a larger dataset can be better utilized for curve fitting.

By plotting the recorded RSSI over the ground truth distance for each sniffer and calculating the path loss model with OLS, each of them can be individually calibrated, with the results depicted in Figure 5.2, where the raw RSSI is shown in blue, and the fitted path loss model is depicted in red.

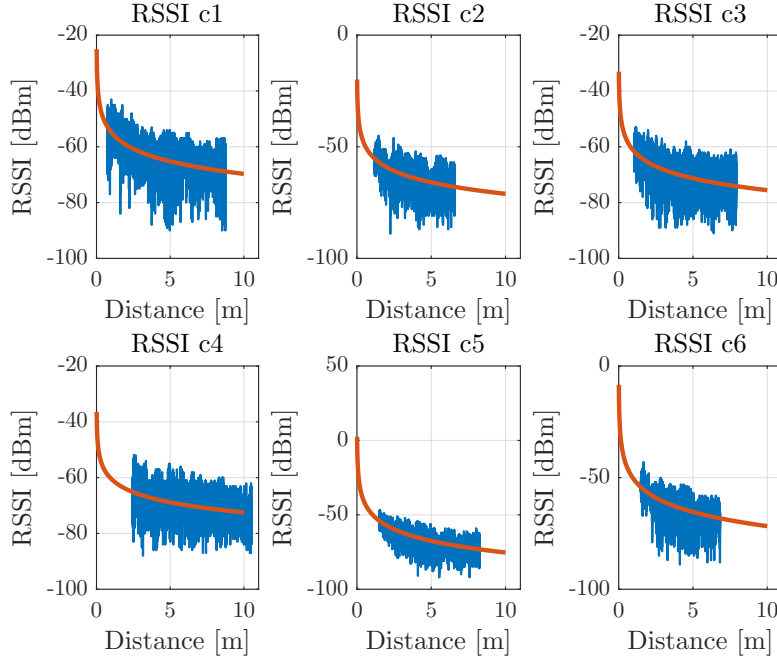


Figure 5.2: Individual calibration of the path loss model of each sniffer using a moving robot.

The fitted path loss coefficients are shown in Table 5.1. As can be seen, the path loss coefficients differ significantly from each other. A single combined path loss model for all sniffers might also work but the error of the distance estimation is expected to rise.

Table 5.1: Coefficients of the individually fitted path loss model for each sniffer.

Sniffer	c_1	c_2	c_3	c_4	c_5	c_6
β_1	-59.5	-54.00	-61.39	-60.44	-49.37	-50.61
β_2	-14.89	-17.07	-14.16	-12.00	-25.93	-21.11

With the knowledge of the path loss model and the ground truth position, the theoretical RSSI can be calculated and compared to the measured RSSI. With this approach, the further processing of the RSSI is shown very intuitively.

5.2.2 Filtering and Distance Estimation

After the calibration, the filtering of the raw RSSI of each sniffer is investigated. The filtering is suggested as a countermeasure against heavily fluctuating RSSI measurements. As mentioned in Section 3.4, the filtering can be realized in two stages. The first stage acts as a downsampler and calculates the average of the specified time window T_g . The second stage builds upon the first stage and calculates a moving average. The decoupling of the two stages is expected to lead to better performance since the filtering

of the second stage and the positioning algorithm can be performed on fewer samples than would be necessary for the first stage.

The RSSI filtering result over time for path S_2 of sniffer c_1 , consisting of the two BLE receivers $c_{1,1}$ and $c_{1,2}$ is depicted in Figure 5.3. The upper figure shows the RSSI and MRSSI for BLE receiver $c_{1,1}$, the middle figure for BLE receiver $c_{1,2}$ and the lower figure for BLE sniffer c_1 .

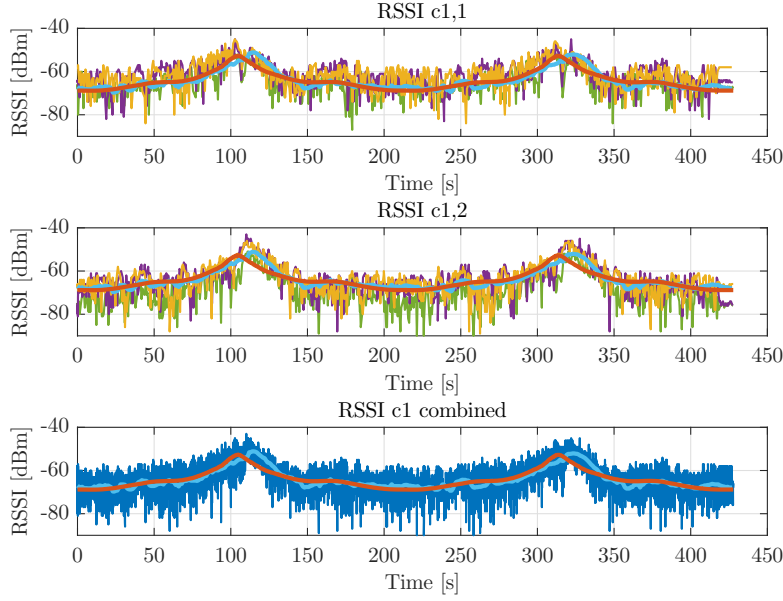


Figure 5.3: Raw, filtered, combined and ground truth RSSI of sniffer c_1 over time of path S_2 using a moving robot.

Additionally, the figures show the MRSSI in light blue (—) and the expected RSSI calculated using the ground truth data in red (—). The lower figure shows all RSSI from both receivers and all channels indicated in (—), the MRSSI and ground truth RSSI are indicated by (—) and (—). While the RSSI are varying by up to 20 dBm between two adjacent measurements, it can be observed, that a good match between the MRSSI and the expected ground truth RSSI is obtained.

Figure 5.4 shows the MRSSI of the sniffers c_i in dark blue (—) and the expected RSSI from the ground truth system in red (—) over time for path S_2 . As shown in Figure 5.4, the measured and the expected RSSIs match for high RSSI amplitudes. The shapes are identical, but the amplitudes sometimes mismatch which is related to multipath propagation. Especially in low RSSI amplitudes, more uncertainty is observable.

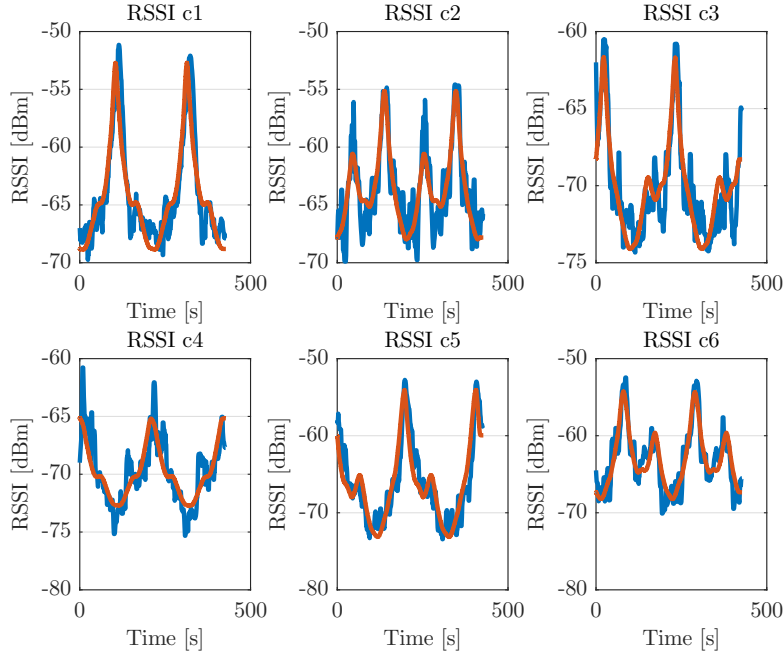


Figure 5.4: Combined and ground truth RSSI of all sniffers c_i over time of path S_2 using a moving robot.

Figure 5.5 shows the estimated distance based on the MRSSI of the sniffers c_i in dark blue (—) and the expected ground truth distance from the ground truth system in red (—) over time for path S_2 . As can be seen in Figure 5.5, the estimated and the expected RSSI match for low distances. As the distance increases, uncertainties due to multipath propagation are more observable.

5.3 Evaluation of the Positioning Performance

To evaluate the positioning performance of a SIRPF for indoor positioning purposes with different sniffer setups and the transition models described in Section 3.5, multiple measurement campaigns are conducted in the indoor environment described in Section 5.1. To rate the positioning accuracy, the root mean square error (RMSE) between the estimated and the ground truth position is used as an error metric.

5.3.1 Positioning Performance of a Moving Robot

The second measurement campaign aims to evaluate the SIRPF positioning algorithm for the mobile robot described in Section 3.7. The robot is programmed to traverse the path S_2 depicted in black in Figure 5.1b with a constant speed of 10 cm/s. The shape of the traversed track is purposely chosen to have no sudden but only smooth changes in its direction. Corners are mostly difficult to track for estimation algorithms and are as well not characteristic for dynamic systems following the principle of inertia.

Based on the acquired path loss model described in Section 5.2.1 and the distance estimates explained in Section 5.2.2, the SIRPF positioning algorithm according to

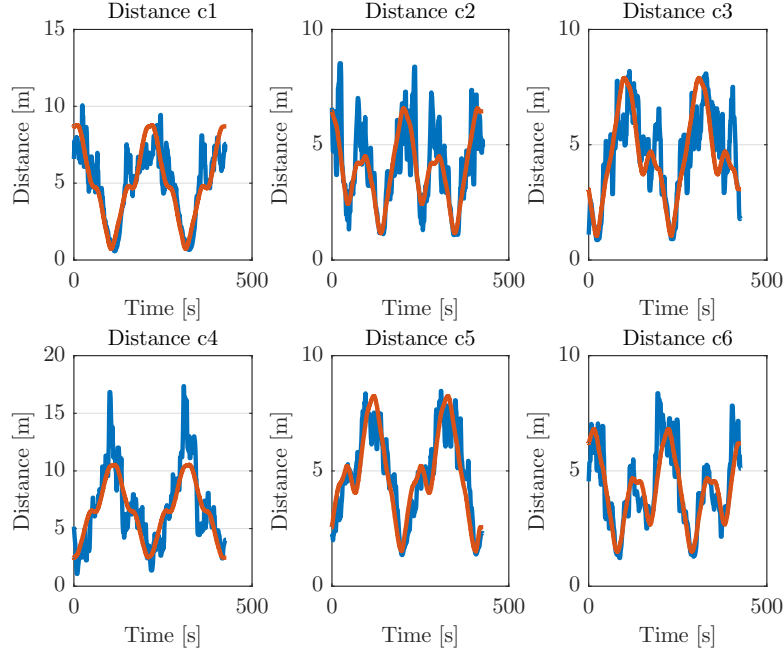


Figure 5.5: Estimated and ground truth distance to all sniffers c_i over time of path S_2 using a moving robot.

Algorithm 2.2 is performed. The parameters $T_g = 0.5$ s for the SIRPF, $\sigma_{q_u} = 0.05$ rad for the Gaussian-Transition-Model and $\sigma_{q_u} = 0.01$ rad, $\sigma_{q_y} = 0.5$ rad/s for the IMU-Transition-Model are determined empirically by multiple positioning runs.

Gaussian-Transition-Model

The first positioning run is performed using the Gaussian-Transition-Model. Figure 5.6 shows the ground truth path indicated in black and the estimated path in light blue using the Gaussian-Transition-Model described in Equation (3.8) for one run of measurement campaign 2 for the mobile robot. The static sniffers c_i are shown in red.

Overall, a good positioning performance can be obtained. The estimated path in light blue matches the ground truth path in black.

Figure 5.7 shows the associated time course of the SIRPF broken down into the individual states. The ground truth data is indicated in blue, the estimated data is depicted in red. The first row shows the data for the x-axis, and the second row for the y-axis. The left column shows the position, and the right the velocity vectors. The ground truth velocity vector is calculated by

$$v_{x,y}(t_k) = \frac{p_{x,y}(t_k) - p_{x,y}(t_{k-1})}{T_g} \quad (5.1)$$

with $T_g = t_k - t_{k-1}$. As there is no sensor providing measurements of the velocity vectors $v_{x,y}(t_k)$ that are used in the Gaussian-Transition-Models, the ground truth velocity vectors are calculated for plotting purposes and to visualize, whether those states are estimated correctly. As shown in Figure 5.7, the estimated position matches the ground

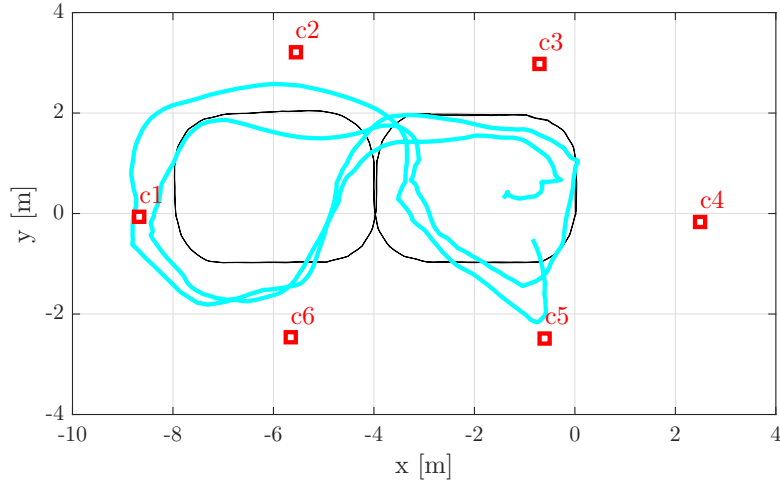


Figure 5.6: SIRPF positioning result of a moving robot using the Gaussian-Transition-Model for path S_2 .

truth position. The quality of the position in x-direction is slightly better than in y-direction. The velocity vectors are estimated without measurements but nevertheless show a good match with the ground truth equivalent but are lagging, which might be the reason for the overshooting of the position estimate.

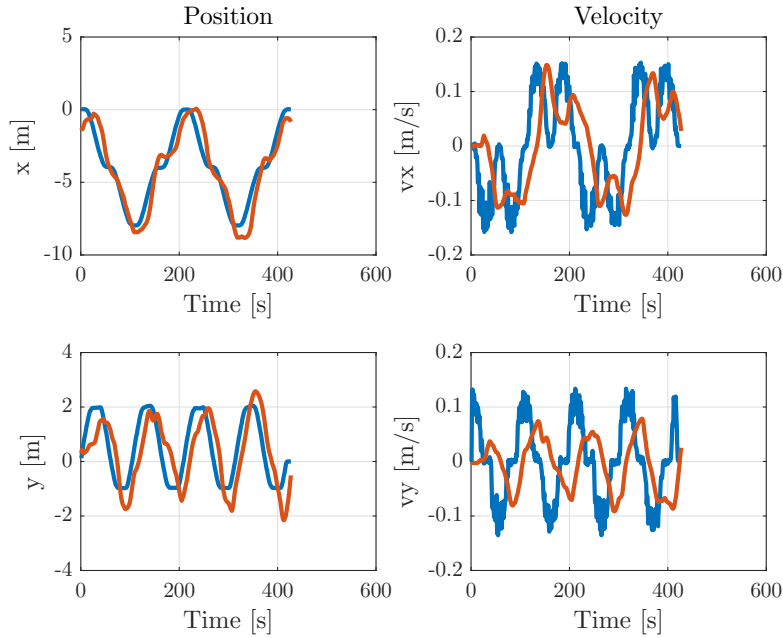


Figure 5.7: SIRPF tracking states of a moving robot using the Gaussian-Transition-Model for path S_2 .

IMU-Transition-Model

The positioning of the SIRPF with the dataset used in Section 5.3.1 is also performed using the IMU-Transition-Model. The IMU-Transition-Model is based on measurements consisting of distance estimates and the yawrate measurement of the mobile device. As the mobile robot carrying the mobile device used in this experiment is equipped with omnidirectional wheels, its orientation and yawrate do not change. However, to investigate whether the IMU-Transition-Model achieves better positioning performance, the heading and yawrate measurements are simulated based on the ground truth data and an initial heading is assumed. The heading $\varphi_{B_b}(t_k)$ can be calculated with

$$\varphi_{B_b}(t_k) = \arctan \left(\frac{p_y(t_k) - p_y(t_{k-1})}{p_x(t_k) - p_x(t_{k-1})} \right). \quad (5.2)$$

The yawrate $\Delta\varphi_{B_b}(t_k)$ is calculated according to Equation (3.15) and is then fused as a measurement in the SIRPF.

Figure 5.8 shows the ground truth path S_2 indicated in black and the estimated path in light blue using the IMU-Transition-Model described in Equation (3.11) for one run of measurement campaign 2 for the moving robot.

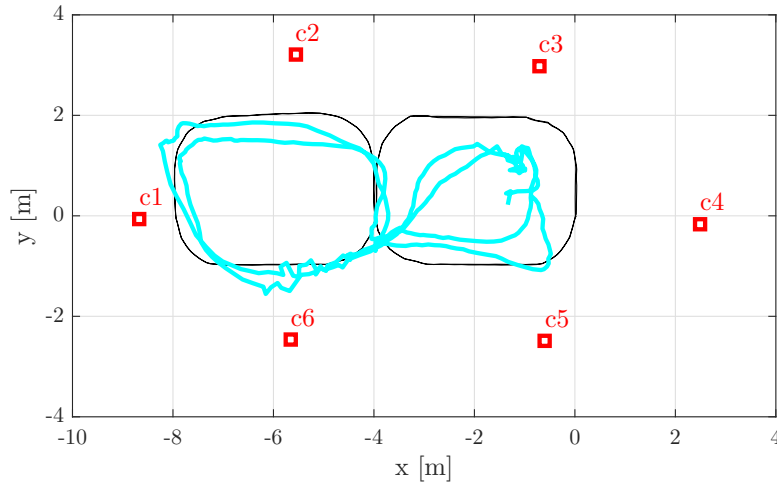


Figure 5.8: SIRPF positioning result of a moving robot using the IMU-Transition-Model for path S_2 .

Figure 5.9 shows the associated time course of the SIRPF broken down into the individual states, where the ground truth data is indicated in blue and the estimated data is depicted in red. The left column depicts the position in the x- and y-direction, and the right column shows the speed vector and the heading. The ground truth speed magnitude can be calculated with

$$v(t_k) = \frac{\sqrt{(p_x(t_k) - p_x(t_{k-1}))^2 + (p_y(t_k) - p_y(t_{k-1}))^2}}{T_g}, \quad (5.3)$$

but is just used for plotting purposes and is not used as a measurement for the SIRPF since a measurement of the speed is not provided by any sensor.

As can be seen in Figure 5.7, the estimated position matches the ground truth better with the IMU-Transition-Model than with the Gaussian-Transition-Model. The speed estimation is sufficient. The heading estimation matches the ground truth heading.

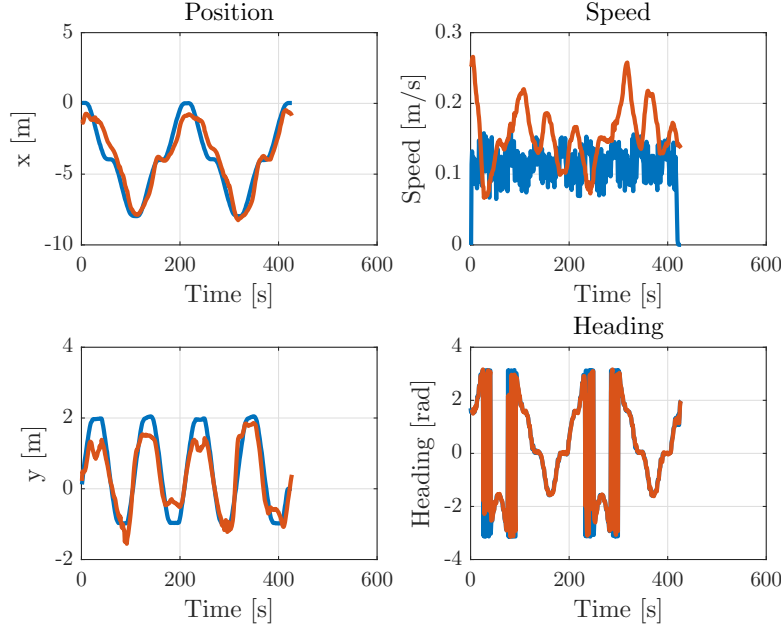


Figure 5.9: SIRPF tracking states of a moving robot using the IMU-Transition-Model for path S_2 .

The positioning for measurement campaign 2 is performed five times for both transition models. Table 5.2 shows the result of the five positioning runs. With the Gaussian-Transition-Model, the RMSE is about 1.1 m. The IMU-Transition-Model showed a RMSE of 0.75 m, which corresponds to an error reduction of 31.8 %.

Table 5.2: RMSE for five runs for measurement campaign 2 with different transition models.

Run	1	2	3	4	5
Gaussian-Transition-Model	1.12 m	1.05 m	1.09 m	1.14 m	1.20 m
IMU-Transition-Model	0.75 m	0.81 m	0.69 m	0.78 m	0.85 m

In Section 3.5, the IMU-Transition-Model is expected to give better performance since the SIRPF has more information about the current target state than without an IMU. The proposed better result using the IMU-Transition-Model is evidenced based on the experiment in measurement campaign 2.

Influence of the Number of Particles

As described in Section 2.2.2, a PF uses set of N_p particles to approximate the a posteriori distribution $p(\mathbf{x}(t_k) \mid \mathbf{z}(t_{1:k}))$. Hence, the positioning performance in terms of

the quality of the estimation is dependent on the number of particles N_p . To evaluate the impact of N_p on the positioning performance, the experiment is conducted for both transition models with $N_p \in \{100, 500, 1000, 1500, 2000, 3000\}$. The results are depicted as CDFs in Figure 5.10.

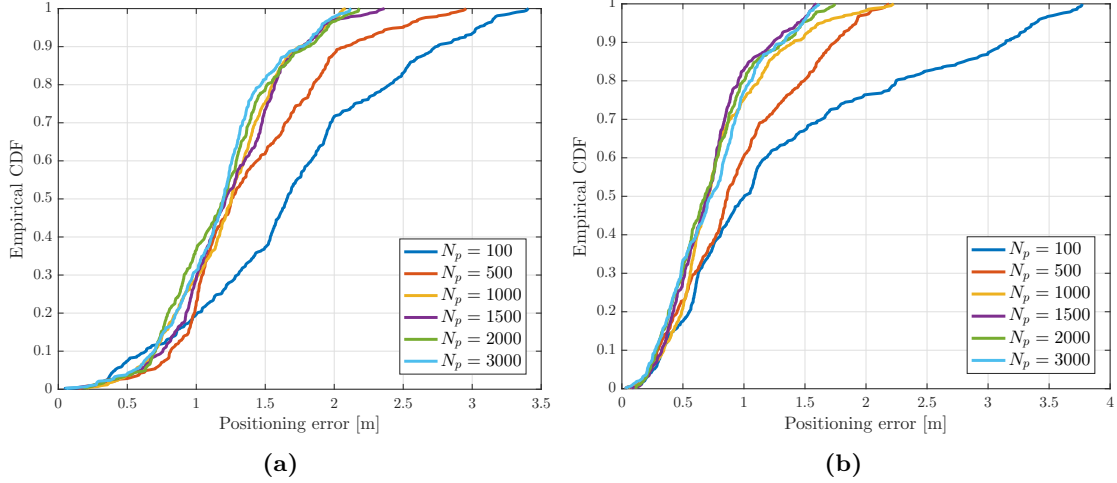


Figure 5.10: Evaluation of the positioning performance of a moving robot for a different number of particles with two transition models.

- (a) CDF of the positioning error for a moving robot using the Gaussian-Transition-Model.
(b) CDF of the positioning error for a moving robot using the IMU-Transition-Model.

As can be seen in both transition models, the distance error decreases as the number of particles increases. Nevertheless, the resulting positioning error converges to a distinct level when $N_p > 1000$. As a result of this, $N_p = 1000$ is used in all further experiments. Having a look at the Algorithms 2.2 and 2.1 showing the SIRPF, a runtime complexity of $\mathcal{O}(N_p)$ is achieved. Hence, in cases where high dynamic response is required, a tradeoff between computational effort and positioning accuracy might be reasonable.

Influence of the Number of Sniffers

After the two transition models are validated and good results are achieved, the dependency of the number of sniffers on the positioning accuracy is investigated. The results for different setups and both transition models are depicted in Figure 5.11. The left column shows the results of the Gaussian-Transition-Model and the right column depicts the results for the IMU-Transition-Model. The ground truth path is indicated in black, the estimated position in light blue, and the sniffer positions are shown in red.

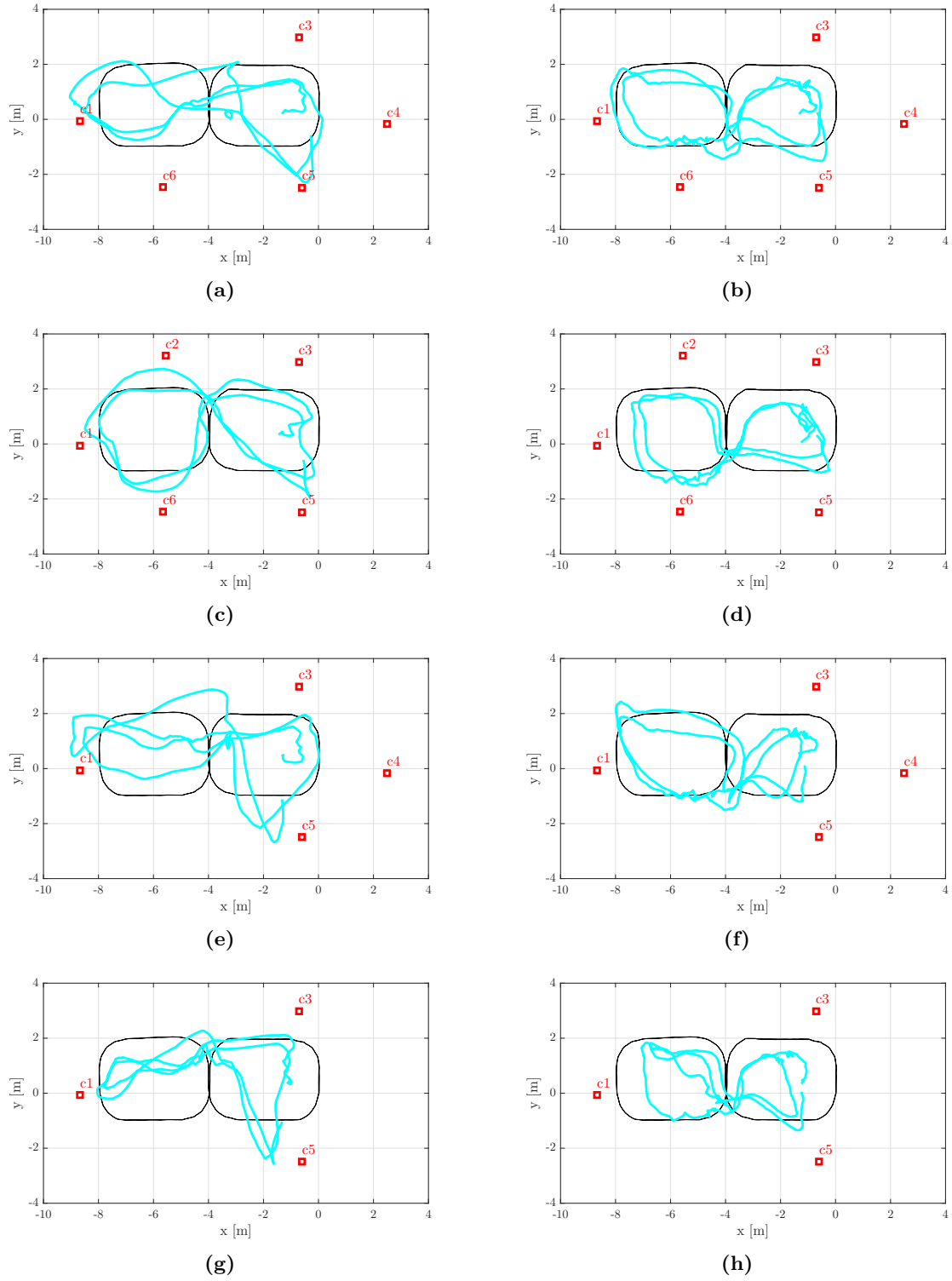


Figure 5.11: Comparison of the positioning result of a moving robot using the Gaussian- and IMU-Transition-Model with different sniffer setups for path S_2 .

As shown in Figure 5.11, the Gaussian-Transition-Model still performs well with five sniffers but the accuracy drops when below five sniffers are in use. With four and three sniffers the shape of the track is not recognizable anymore. In contrast to that, the IMU-Transition-Model still has remarkable performance when only three sniffers are utilized where the shape is still recognizable but the accuracy drops as well.

The achieved RMSE for Gaussian-Transition-Model is depicted in Table 5.3.

Table 5.3: RMSE for measurement campaign 2 with the Gaussian-Transition-Model.

Setup	a)	c)	e)	g)
Gaussian-Transition-Model	1.18 m	1.01 m	1.68 m	1.66 m

Table 5.4 shows the RMSE for the IMU-Transition-Model.

Table 5.4: RMSE for measurement campaign 2 with the IMU-Transition-Model.

Setup	b)	d)	f)	h)
IMU-Transition-Model	0.79 m	0.62 m	0.87 m	0.94 m

The CDFs of the positioning error for the sniffer setups evaluated in Figure 5.11 with the two motion models are depicted in Figure 5.12.

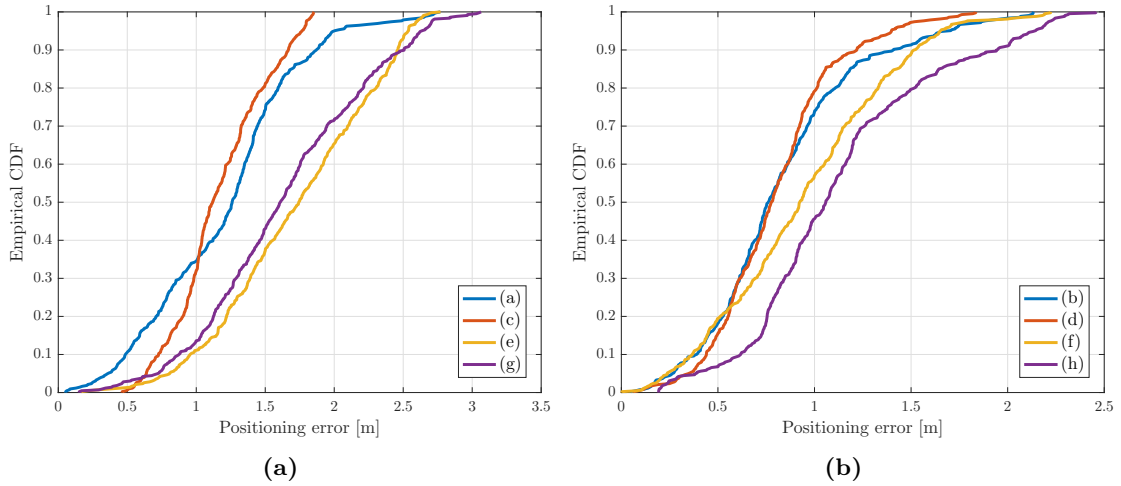


Figure 5.12: Evaluation of the positioning performance of a moving robot for a different number of sniffers with two transition models.

- (a) CDF of the positioning error for a moving robot using the Gaussian-Transition-Model.
- (b) CDF of the positioning error for a moving robot using the IMU-Transition-Model.

Comparing the CDFs of Figure 5.12a using the Gaussian-Transition-Model with Figure 5.12b using the IMU-Transition-Model, the IMU-Transition-Model outperforms the Gaussian-Transition-Model in terms of the total positioning error.

Influence of the Time Window T_g

As explained in Section 3.4, the distance and position estimation of the SIRPF run on a fixed sampled time grid t_k , with $T_g = t_k - t_{k-1}$ denoting the time between two adjacent time stamps. The bigger T_g is chosen, the more RSSI measurements are used to determine the MRSSI on the sniffers and the more the RSSI data is smoothed. To evaluate the impact of T_g on the positioning performance, the experiment is conducted for both transition models with $T_g \in \{0.2s, 0.5s, 1.0s, 2.0s\}$. The results are depicted as CDFs in Figure 5.13.

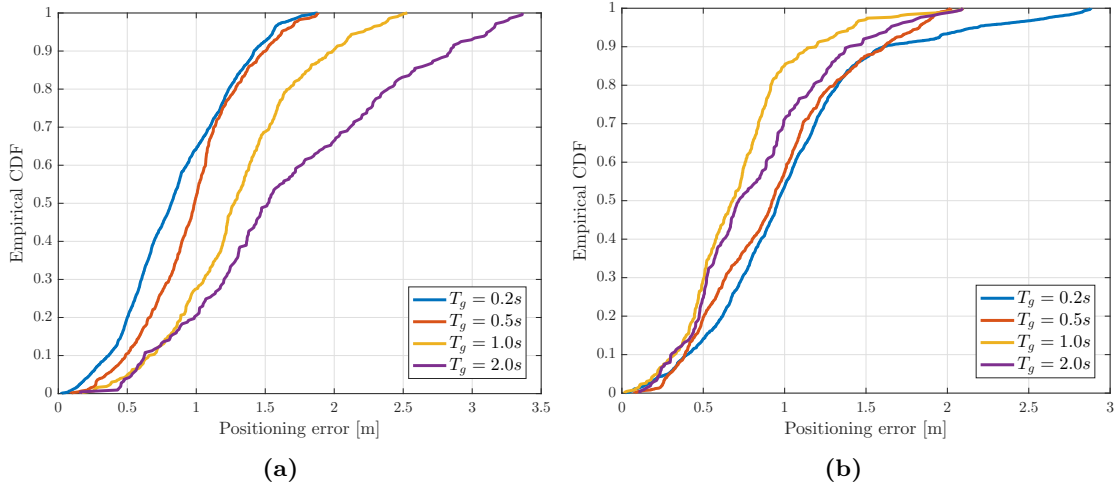


Figure 5.13: Evaluation of the positioning performance of a pedestrian with two transition models and different T_g .

(a) CDF of the positioning error for a robot using the Gaussian-Transition-Model.

(b) CDF of the positioning error for a robot using the IMU-Transition-Model.

Figure 5.13a shows better positioning performance with lower T_g for the Gaussian-Transition-Model, whereas the Figure 5.13b shows similar positioning performance for all T_g for the IMU-Transition-Model.

5.3.2 Positioning Performance of a Pedestrian

After measurement campaign 2 shows remarkable performance for a moving robot, a third measurement campaign is conducted in the same area to obtain the positioning performance for a pedestrian holding the mobile device. The path S_3 to be traversed is depicted in black in Figure 5.14b and has a similar shape to the path S_2 of the robot in measurement campaign 2.

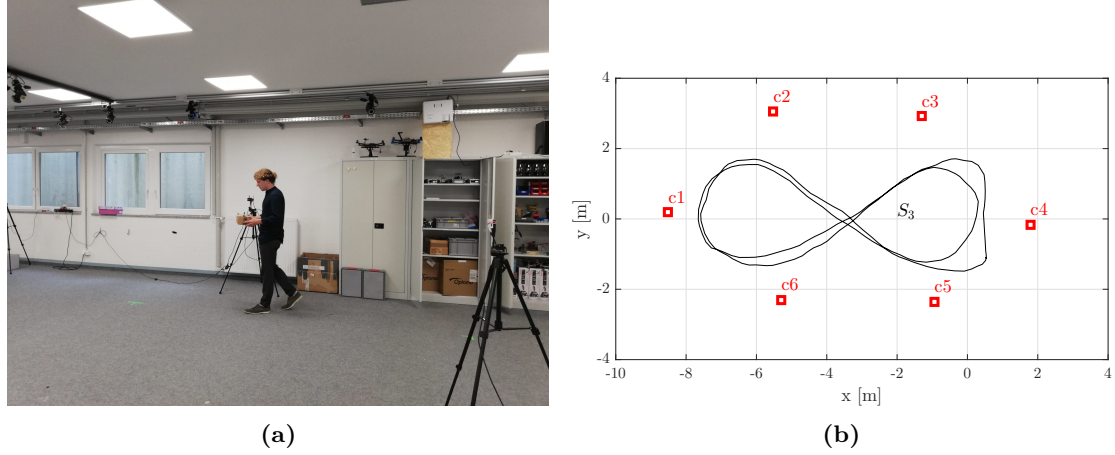


Figure 5.14: Indoor environment for measurement campaign 3.

(a) Indoor measurement setup with a pedestrian and the BLE sniffers c_i mounted on tripods for measurement campaign 3.

(b) Ground truth view of the indoor environment with the traversed path S_3 for a pedestrian and the sniffers c_i .

To have a ground truth reference to quantify the positioning error, the mobile device is mounted in a box, which is tracked by the optical motion capture system.

The positioning error is expected to increase in comparison to measurement campaign 2. Since the mobile device is carried, the human body acts as an absorber of the BLE signal and attenuates the RSSI. As a result, errors in the estimated distances might lead to worse positioning performance.

Moreover, the orientation of the mobile device is changing as the path is traversed. The orientation is determined by the mobile device itself using IMU measurements and a fusion algorithm. The fusion result is embedded in the BLE advertisements. The positioning performance is thus relying on the accuracy of the orientation calculation of the mobile device.

As the mobile device's orientation changes, the orientation of its BLE antenna changes as well. In measurement campaign 2 the orientation of the mobile's BLE antenna is constant since the robot is not changing its orientation to traverse the path due to the omnidirectional wheels. Hence, it is expected, that multipath propagation now has a bigger impact on the result than in measurement campaign 2.

Accuracy of the Mobile Device's Heading Determination

The first task of measurement campaign 3 is to investigate the accuracy of the mobile's heading estimation. To compare it with the ground truth heading, it has to be converted according to the heading alignment described in Section 4.4.3.

In Figure 5.15 the ground truth heading is indicated in red. Since there is no movement in the intervals $[0, 15]$ s and $[105, 140]$ s the ground truth heading is invalid here and flips. The calculated heading on the mobile device is indicated in blue where the initial angle is $\pi/2$ and the data is free of drift. As can be seen, in the dynamic phases

both angles match and the heading alignment works.

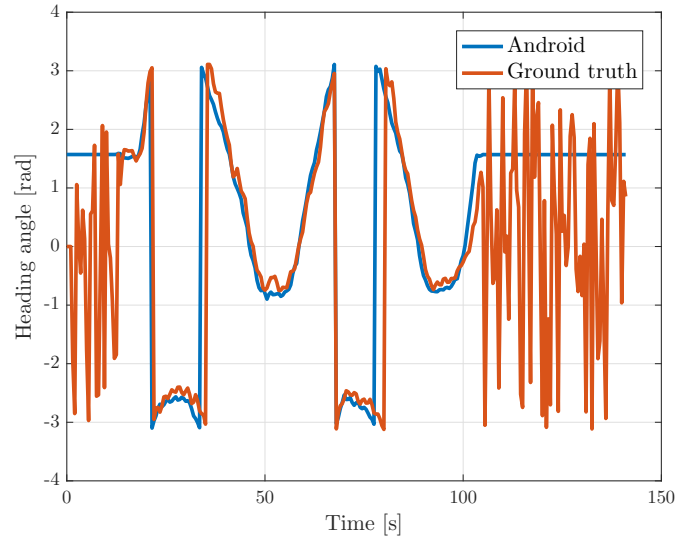


Figure 5.15: Comparison of the ground truth heading and the heading estimation of the mobile device after the heading alignment.

The positioning is then performed again for both transition models with different setups regarding the number of sniffers, where the results are depicted in Figure 5.16. The left column shows the results of the Gaussian-Transition-Model and the right column depicts the results for the IMU-Transition-Model. The ground truth path is indicated in black, the estimated position in light blue, and the sniffer positions are shown in red.

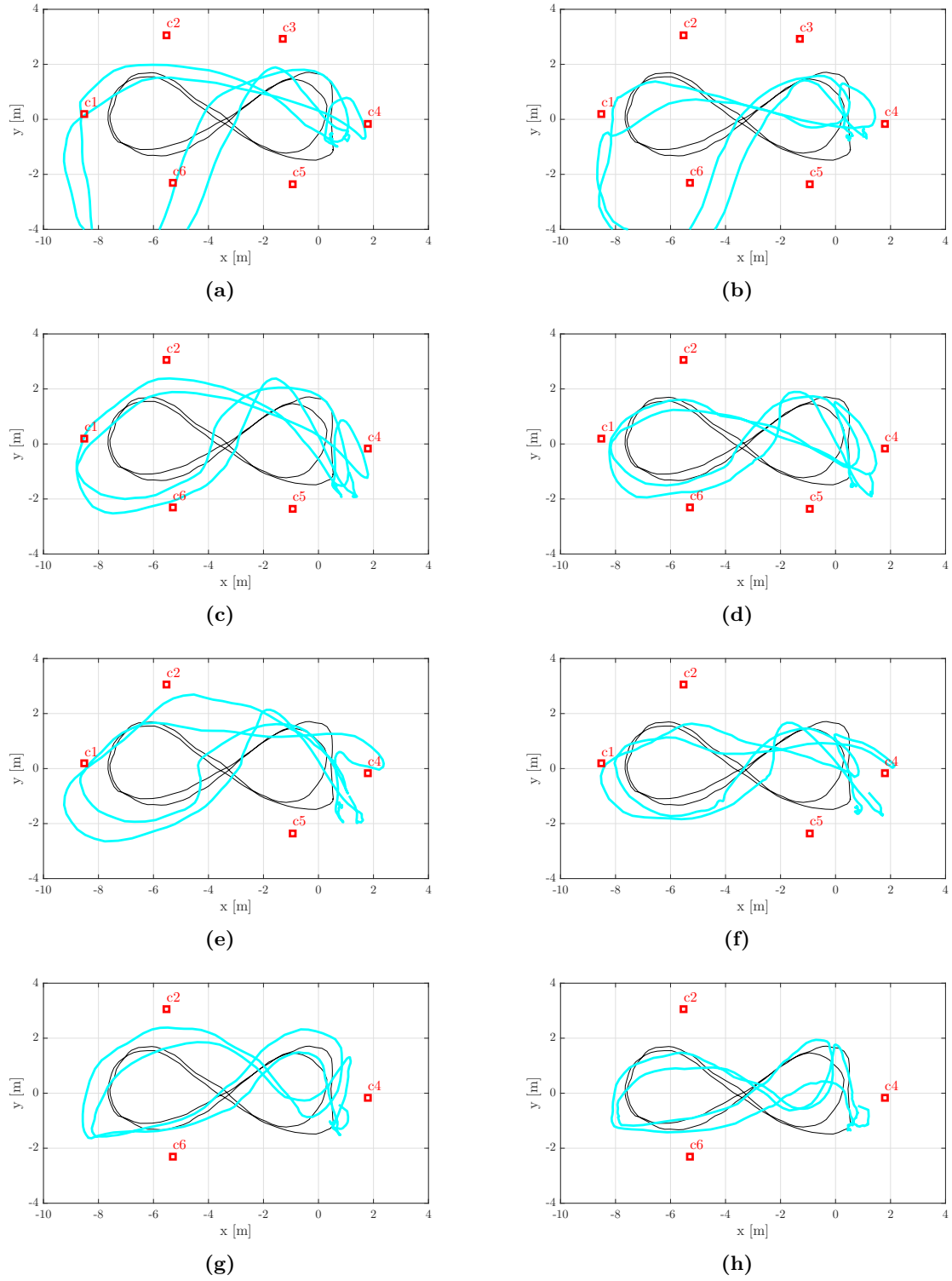


Figure 5.16: Comparison of the positioning result of a pedestrian using the Gaussian- and IMU-Transition-Model with different sniffer setups for path S_3 .

As can be seen in Figure 5.16, both models perform better when fewer sniffers are utilized. In general the results show a better positioning result for the IMU-Transition-Model. The achieved RMSE for the Gaussian-Transition-Model is depicted in Table 5.5 and for the IMU-Transition-Model in Table 5.6.

Table 5.5: RMSE for measurement campaign 3 with the Gaussian-Transition-Model.

Setup	a)	c)	e)	g)
Gaussian-Transition-Model	2.05 m	1.16 m	1.40 m	0.99 m

Table 5.6: RMSE for measurement campaign 3 with the IMU-Transition-Model.

Setup	b)	d)	f)	h)
IMU-Transition-Model	1.3 m	0.89 m	1.08 m	0.94 m

In comparison to the Gaussian-Transition-Model, the IMU-Transition-Model reduces the positioning error by an average of 25 % among the different sniffer setups.

The CDFs of the positioning error for the sniffer setups evaluated in Figure 5.16 with the two motion models are depicted in Figure 5.17.

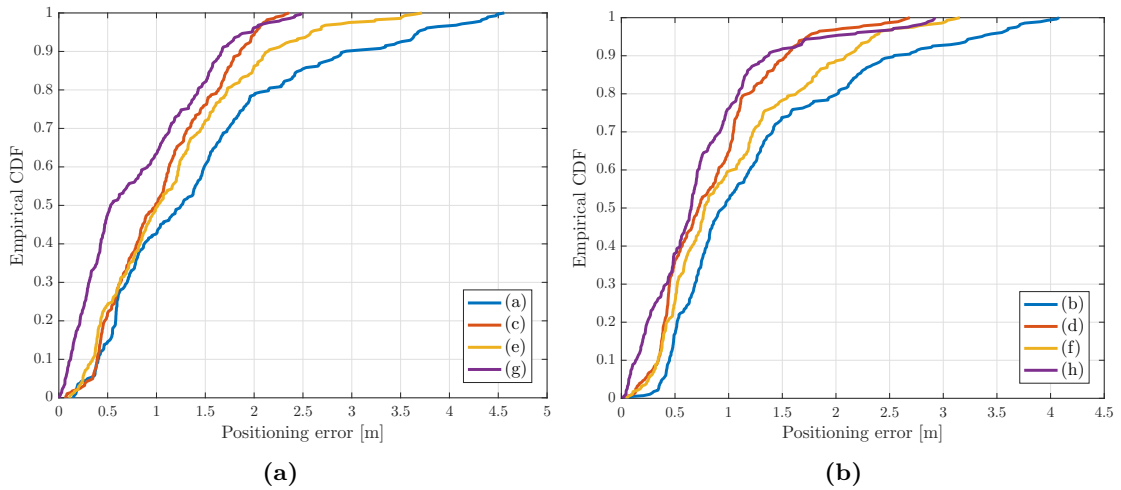


Figure 5.17: Evaluation of the positioning performance of a pedestrian for a different number of sniffers with two transition models.

(a) CDF of the positioning error of a pedestrian using the Gaussian-Transition-Model.

(b) CDF of the positioning error of a pedestrian using the IMU-Transition-Model.

Comparing the CDFs of Figure 5.17a using the Gaussian-Transition-Model with Figure 5.17b using the IMU-Transition-Model, both models perform similar in terms of total positioning error. In contrast to measurement campaign 2 described in Section 5.3.1, the positioning result obviously depends more on the sniffer setup than on the number of the sniffers.

Influence of the Sniffers Setup

In Figure 5.16 the achievable positioning performance with the two transition models is depicted. Especially worth mentioning is the difference between the Figures 5.16b and 5.16d for the IMU-based approach with the same data. In Figure 5.16d the sniffer c_3 is not considered in the SIRPF and the positioning error is reduced from 1.3 m in Figure 5.16b to 0.9 m in Figure 5.16d, which corresponds to a reduction of 31.5 %. This case is further investigated.

To acquire a better understanding, the RSSIs and the estimated distances are first focused on. In Figure 5.18 the time course of the MRSSIs are depicted in blue and the expected ground truth RSSIs are shown in red. As can be seen, the MRSSIs match for every sniffer, except for sniffer c_3 . The MRSSI obtained by sniffer c_3 differs tremendously from the ground truth especially in the intervals $[27, 49]$ s and $[72, 93]$ s. The maximum difference is approximately -8 dBm. The error in the MRSSI is expected to originate from multipath propagation.

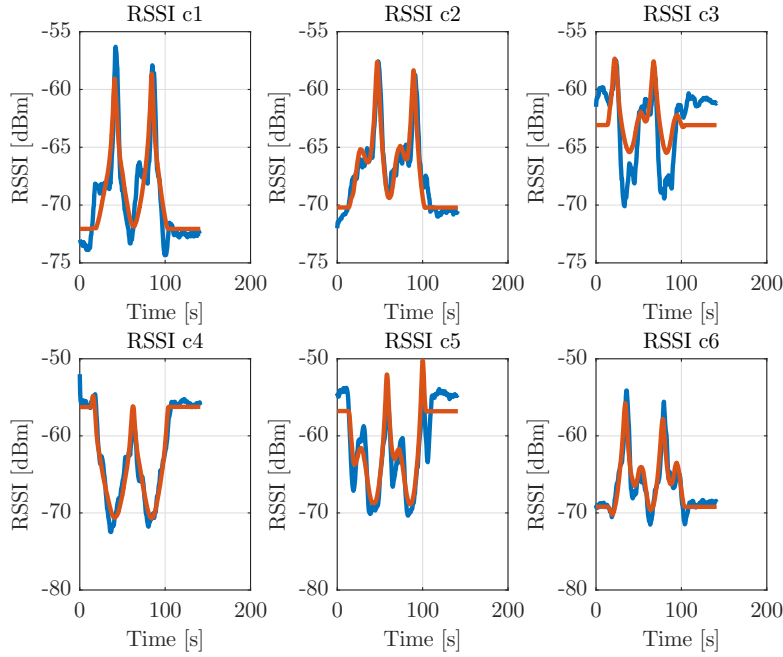


Figure 5.18: Combined and ground truth RSSI of all sniffers c_i over time of path S_3 for a pedestrian.

Figure 5.19 shows the time course of the estimated distances for every sniffer. In the mentioned intervals, the sniffer c_3 shows a maximum distance estimation error of approximately 13 m. The enormous distance error of sniffer c_3 is expected to cause problems in the positioning.

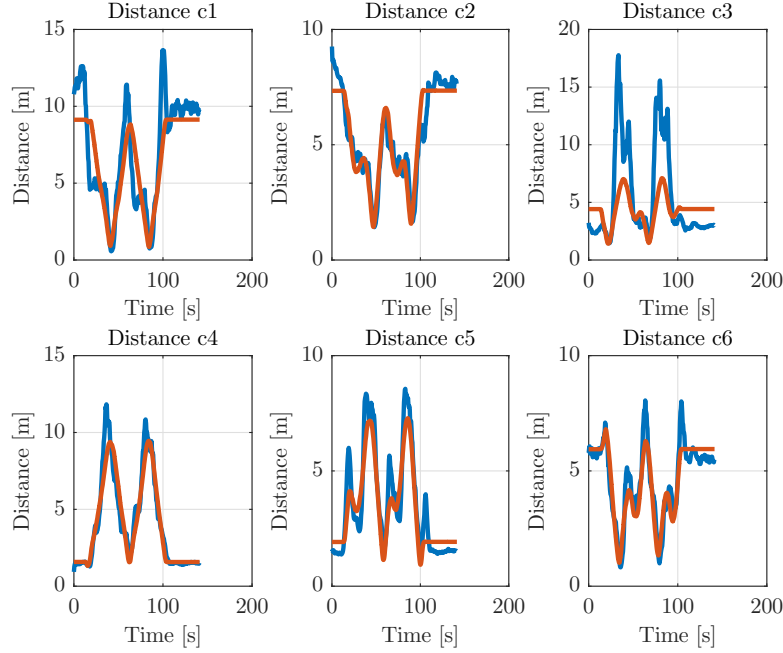


Figure 5.19: Estimated and ground truth distance to all sniffers c_i over time of path S_3 for a pedestrian.

The phenomenon of multipath propagation can as well be observed in measurement campaign 2, but not to this extent. There the multipath mitigation could be compensated with the approach of two closely spaced BLE receivers and a pre-processing of the RSSI. A disturbance of this magnitude cannot be filtered with the proposed filtering approach.

Having a look at measurement campaign 2 in Section 5.3.1, the setup is nearly identical. Sniffer c_3 is utilized but multipath mitigation is barely present. The multipath error in measurement campaign 3 on sniffer c_3 is supposed to be caused by two reasons. As can be seen in Figure 5.1a, sniffer c_3 is placed in front of a metal-built radiator, which is not the case for any other sniffer. The radiator might act as a perfect reflector to the BLE signal when it perpendicularly hits the surface of the radiator. In measurement campaign 2, the mobile device is also not shadowed by a human body. Both aspects might lead to the severe multipath impact.

To test the proposed dynamic weighting, the experiment of measurement campaign 3 described in Section 5.3.2 is conducted again for a pedestrian traversing the path S_3 . In this experiment, sniffer c_3 is considered for as measurement in the SIRPF as depicted in Figure 5.14. The dynamic weighting is implemented in the SIRPF according to Equation (3.19) and the positioning is performed for both transition models where the result is depicted in Figure 5.20.

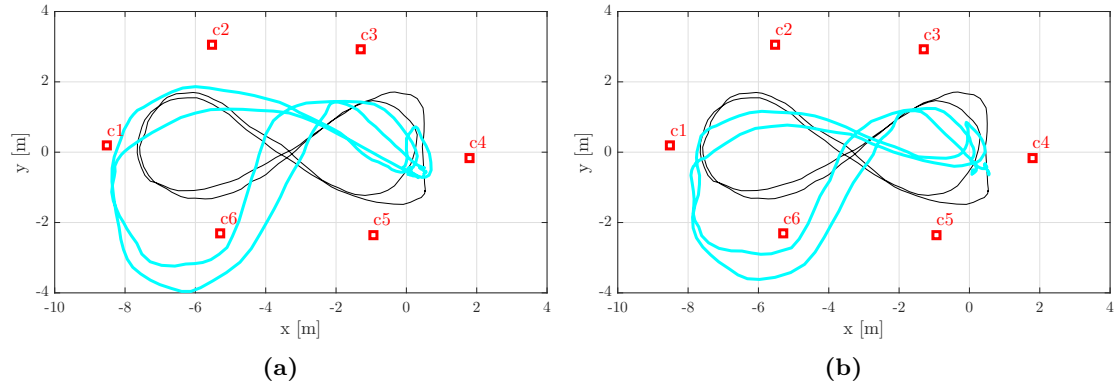


Figure 5.20: Comparison of the positioning result of a pedestrian using dynamic weighting with two transition models for path S_3 .

(a) Positioning result using the Gaussian-Transition-Model with dynamic weighting.

(b) Positioning result using the IMU-Transition-Model with dynamic weighting.

Comparing Figure 5.20a with 5.16a and Figure 5.20b with 5.16b, a clear improvement is achieved by dynamic weighting. A direct comparison of normal and dynamic weighting for both transition models is given by the CDFs in Figure 5.21.

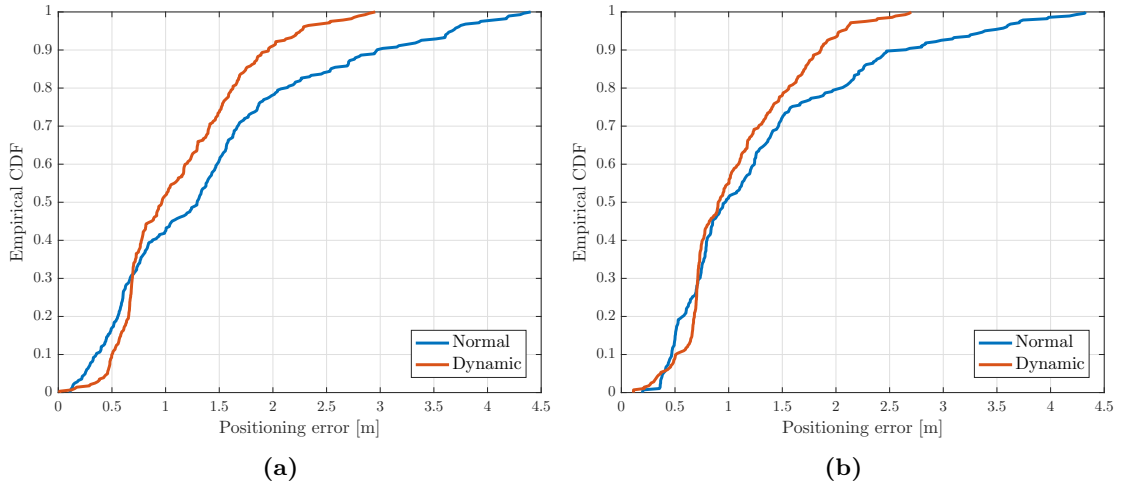


Figure 5.21: Evaluation of the positioning performance of a pedestrian using dynamic weighting with two transition models.

(a) CDF of the positioning error for a pedestrian using the Gaussian-Transition-Model with dynamic weighting.

(b) CDF of the positioning error for a pedestrian using the IMU-Transition-Model with dynamic weighting.

As shown in this experiment, the implementation of dynamic weighting is one of the most effective countermeasures against strongly non-linear fluctuations and drastically enhances the positioning accuracy.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, a BLE-based IPS for mobile devices was developed. The system was realized using a server-based approach where the position estimation is performed on a server but not on the mobile device itself. Based on the procedure of advertising and scanning, low-cost USB dongles are programmed as static BLE receivers providing RSSI measurements from a mobile device acting as a BLE beacon. The RSSIs of the BLE receivers are processed on a server, which smooths the RSSIs over time. The positioning utilizes a ranging-based approach, where the distances between the mobile device and multiple static BLE sniffers is used to gather knowledge about the position of the mobile device.

To estimate the distance between the mobile device and static BLE sniffers, the log-distance path loss model was used as a radio propagation model, where the relationship between RSSI values and distance is modeled by a logarithmic relationship. The parameters of this model were determined by minimizing the sum of squared errors between the measured RSSI values and the ground-truth RSSI values using OLS. The ground truth data was obtained using a high-precision optical motion capture system.

The RSSI is subject to highly non-linear and non-Gaussian multipath influences from indoor environments. To still obtain a position estimate, a Bayesian filter was investigated, which was implemented as a SIRPF. To further increase the positioning accuracy, different countermeasures were applied:

- Each BLE sniffer is equipped with multiple closely spaced BLE receivers to reduce small-scale fading caused by multipath propagation.
- The raw RSSI is averaged over multiple channels and over multiple receivers equipped on each sniffer.
- The path loss model of each sniffer is calibrated individually to provide best possible distance estimation.
- The fusion of inertial data obtained by the mobile device's IMU is utilized as another measurand to provide better situation awareness to the SIRPF.
- The weighting of the SIRPF is implemented using a dynamic approach to prioritize BLE sniffers with higher RSSI measurements.

The proposed concept was implemented in a Matlab model and also as an embedded IPS based on a server-based architecture in C++ with multiple Raspberry Pi 4, providing live positioning functionality. Evaluations based on indoor measurements show, that accurate positioning based on a server-based BLE positioning system is feasible. Especially, a positioning error with a mean RMSE of 0.75 m for a mobile robot and a mean RMSE of 1.1 m for a walking pedestrian in an indoor environment was obtained.

6.2 Future Work

The work on the BLE-based IPS is planned to be continued. In this thesis, only 2D scenarios are investigated. In future works, also 3D positioning might be interesting to identify not just the position in a room but also identify the floor. Therefore a bigger measurement campaign is necessary.

Another topic is the tracking of multiple targets, where in this thesis only one target is tracked. A SIRPF might also work for applications with multiple target tracking, resulting in more than one particle swarm. It would have to be investigated, how multiple particle swarms can be distinguished from each other or if multiple SIRPFs are necessary. The number of particles would probably also have to be increased, resulting in a higher computational effort.

Another important aspect may be considering other types of transition models. In this thesis, transition models similar to vehicular motions like driving are investigated, where for pedestrians other models exist. These models often rely on detecting steps also based on inertial measurement data. By implementing a simultaneous localization and mapping system, the BLE-based positioning might help an IMU-based system to periodically recalibrate a possible drift. Combining a step detection and simultaneous mapping approach with BLE-based positioning might be a promising technique for indoor position tracking of pedestrians.

The BLE-based IPS developed in this thesis can as well be used as a backup system for already existing IPS solutions that rely on other technologies like UWB. The other way round is also possible if mobile devices provided better access to UWB functionality. In this case, the positioning accuracy might be enhanced dramatically and probably allow cm precision.

This thesis presents a range-based approach to determine the indoor position of a mobile device. Based on the RSSI, the distance to fixed reference points based on a radio propagation model is calculated. It is also possible to determine a position using an angular approach. This feature, also referred to as direction finding, was introduced in Bluetooth standard 5.1 in 2019. However, the direction-finding functionality is not yet supported by the operating systems of mobile devices. The benefit and the impact on the accuracy gained by including the direction-finding feature would have to be investigated as well in future works.

List of Figures

2.1	Principle of the trilateration method to estimate the position relative to a known coordinate system.	4
2.2	Physical layer of BLE spectrum with overlapped Wi-Fi channels.	5
2.3	Principle of a BLE advertiser and scanner.	6
2.4	Structure of a BLE advertising packet split up into individual parts. . .	7
2.5	Multipath phenomena in an indoor environment showing reflection, attenuation, diffraction, and scattering.	8
2.6	Principle of a first order HMM where the current state is observed using a measurement and only depends on the previous state.	10
2.7	Systematic resampling used as a measure against degeneracy in PFs. . .	14
2.8	One iteration of a SIRPF including the steps: Drawing, weighting, and resampling.	16
3.1	Comparison of two possible system architectures implemented as device-based and server-based IPS.	18
3.2	nRF54840 Dongle [18] used as a BLE receiver to record BLE packets advertised from a mobile device acting as BLE beacon.	19
3.3	Indoor environment for measurement campaign 1 to acquire the path loss model.	20
3.4	Indoor environment for measurement campaign 1 to acquire the path loss model.	21
3.5	Two staged RSSI filtering of the raw RSSI to obtain less fluctuations for the distance estimation.	23
3.6	Illustration of the Gaussian-Transition-Model.	25
3.7	Illustration of the IMU-Transition-Model.	27
4.1	Flowchart of the nRF52840 Dongle firmware implementation.	33
4.2	Screenshot of the Android application displaying the sensor values of the inertial sensors and the fusion result.	35
4.3	Server-based system architecture of the embedded IPS with all data flows. .	37
4.4	Class diagram of the server main program.	40
5.1	Indoor environment for measurement campaign 2.	42
5.2	Individual calibration of the path loss model of each sniffer using a moving robot.	44

5.3	Raw, filtered, combined and ground truth RSSI of sniffer c_1 over time of path S_2 using a moving robot.	45
5.4	Combined and ground truth RSSI of all sniffers c_i over time of path S_2 using a moving robot.	46
5.5	Estimated and ground truth distance to all sniffers c_i over time of path S_2 using a moving robot.	47
5.6	SIRPF positioning result of a moving robot using the Gaussian-Transition-Model for path S_2	48
5.7	SIRPF tracking states of a moving robot using the Gaussian-Transition-Model for path S_2	48
5.8	SIRPF positioning result of a moving robot using the IMU-Transition-Model for path S_2	49
5.9	SIRPF tracking states of a moving robot using the IMU-Transition-Model for path S_2	50
5.10	Evaluation of the positioning performance of a moving robot for a different number of particles with two transition models.	51
5.11	Comparison of the positioning result of a moving robot using the Gaussian- and IMU-Transition-Model with different sniffer setups for path S_2	52
5.12	Evaluation of the positioning performance of a moving robot for a different number of sniffers with two transition models.	53
5.13	Evaluation of the positioning performance of a moving robot with two transition models and different T_g	54
5.14	Indoor environment for measurement campaign 3.	55
5.15	Comparison of the ground truth heading and the heading estimation of the mobile device after the heading alignment.	56
5.16	Comparison of the positioning result of a pedestrian using the Gaussian- and IMU-Transition-Model with different sniffer setups for path S_3	57
5.17	Evaluation of the positioning performance of a pedestrian for a different number of sniffers with two transition models.	58
5.18	Combined and ground truth RSSI of all sniffers c_i over time of path S_3 for a pedestrian.	59
5.19	Estimated and ground truth distance to all sniffers c_i over time of path S_3 for a pedestrian.	60
5.20	Comparison of the positioning result of a pedestrian using dynamic weighting with two transition models for path S_3	61
5.21	Evaluation of the positioning performance of a pedestrian using dynamic weighting with two transition models.	61

Abbreviations

2D	two-dimensional
3D	three-dimensional
API	Application Programming Interface
BLE	Bluetooth Low Energy
CDF	cumulative distribution function
CRC	cyclic redundancy check
DHCP	Dynamic Host Configuration Protocol
EKF	Extended Kalman filter
GNSS	global navigation satellite system
HMM	hidden Markov model
HIL	hardware-in-the-loop
ID	identification number
IP	Internet Protocol
ISM	industrial, scientific and medical
IMU	inertial measurement unit
IPS	indoor positioning system
KF	Kalman filter
MAP	maximum a posteriori
MMSE	minimum mean square error
MRSSI	mean received signal strength indicator
NTP	Network Time Protocol
OLS	ordinary least squares

PDF	probability density function
PDU	protocol data unit
PF	particle filter
RFID	radio frequency identification
RMSE	root mean square error
RSS	received signal strength
RSSI	received signal strength indicator
SIRPF	sequential importance resampling particle filter
SISPF	sequential importance sampling particle filter
TCP	Transmission Control Protocol
UTC	Coordinated Universal Time
UKF	Unscented Kalman Filter
UWB	ultra-wideband
USB	Universal Serial Bus
UDP	User Datagram Protocol
WLAN	wireless local area network

Appendix A

Supplementary Materials

List of supplementary data submitted to the degree-granting institution for archival storage (in ZIP format).

A.1 PDF Files

Path: /

thesis.pdf Master thesis (complete document)

References

Literature

- [1] M. S. Arulampalam et al. “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”. *IEEE Transactions on Signal Processing* 50.2 (2002), pp. 174–188 (cit. on pp. 9, 11, 16).
- [2] Imran Ashraf, Soojung Hur, and Yongwan Park. “Smartphone Sensor Based Indoor Positioning: Current Status, Opportunities, and Future Challenges”. *Electronics* 9 (May 2020), p. 891. DOI: 10.3390/electronics9060891 (cit. on pp. 3, 4, 27).
- [3] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Electronic and electrical engineering. Wiley, 2001 (cit. on p. 24).
- [4] Olivier Cappe, Simon J. Godsill, and Eric Moulines. “An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo”. *Proceedings of the IEEE* 95.5 (2007), pp. 899–924. DOI: 10.1109/JPROC.2007.893250 (cit. on pp. 9, 10).
- [5] Zhe Chen. “Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond”. *Statistics* 182 (Jan. 2003). DOI: 10.1080/02331880309257 (cit. on pp. 9, 16).
- [6] Abdullahi Daniyan, Yu Gong, and Sangarapillai Lambbotharan. “An improved resampling approach for particle filters in tracking”. In: *2017 22nd International Conference on Digital Signal Processing (DSP)*. 2017, pp. 1–5. DOI: 10.1109/ICD SP.2017.8096095 (cit. on p. 14).
- [7] Pavel Davidson and Robert Piché. “A Survey of Selected Indoor Positioning Methods for Smartphones”. *IEEE Communications Surveys & Tutorials* 19.2 (2017), pp. 1347–1370. DOI: 10.1109/COMST.2016.2637663 (cit. on pp. 1, 3, 4).
- [8] Ramsey Faragher and Robert Harle. “Location Fingerprinting With Bluetooth Low Energy Beacons”. *IEEE Journal on Selected Areas in Communications* 33.11 (2015), pp. 2418–2428. DOI: 10.1109/JSAC.2015.2430281 (cit. on p. 5).
- [9] H.T. Friis. “A Note on a Simple Transmission Formula”. *Proceedings of the IRE* 34.5 (May 1946), pp. 254–256. DOI: 10.1109/JRPROC.1946.234568 (cit. on p. 7).

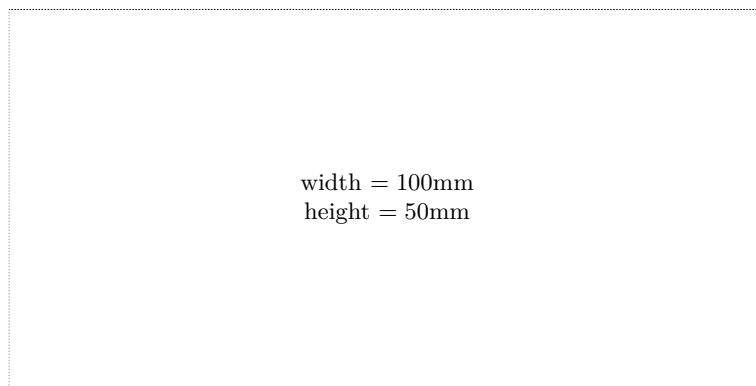
- [10] Christian Gentner, Daniel Günther, and Philipp H. Kindt. “Identifying the BLE Advertising Channel for Reliable Distance Estimation on Smartphones”. *IEEE Access* 10 (2022), pp. 9563–9575. DOI: 10.1109/ACCESS.2022.3140803 (cit. on pp. 17, 18, 34).
- [11] Christian Gentner, Philipp Hager, and Markus Ulmschneider. “Server based BLE Positioning using RSS Measurements”. 2023. In press (cit. on pp. 22, 24, 28, 29).
- [12] R.W. Heath. *Introduction to Wireless Digital Communication: A Signal Processing Perspective*. Pearson Education, 2017, pp. 313–318. URL: <https://books.google.at/books?id=DKGYDgAAQBAJ> (cit. on pp. 7, 8).
- [13] Raspberry Pi (Trading) Ltd. *Datasheet Raspberry Pi 4 Model B*. <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf>. 2019 (cit. on p. 36).
- [14] Simone Ludwig. “Optimization of Control Parameter for Filter Algorithms for Attitude and Heading Reference Systems”. In: July 2018, pp. 1–8. DOI: 10.1109/CEC.2018.8477725 (cit. on pp. 27, 28, 34).
- [15] D.L. Mills. “Internet time synchronization: the network time protocol”. *IEEE Transactions on Communications* 39.10 (1991), pp. 1482–1493. DOI: 10.1109/26.103043 (cit. on pp. 22, 36).
- [16] Branko Ristic, Sanjeev Arulampalam, and Neil J. Gordon. “Beyond the Kalman Filter: Particle Filters for Tracking Applications”. In: Artech House, 2003 (cit. on pp. 10–13, 15, 16).
- [17] Javier Rodas et al. “Multiple Antennas Bluetooth System for RSSI Stabilization”. In: Sept. 2007. DOI: 10.1109/ISWCS.2007.4392421 (cit. on p. 22).
- [18] Nordic Semiconductor. *nRF52840 Product Specification v1.7*. https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.7.pdf. 2021 (cit. on p. 19).
- [19] Bluetooth SIG. *Bluetooth Core Specification 5.3*. <https://www.bluetooth.com/specifications/specs/core-specification-5-3/>. 2021 (cit. on pp. 4–7).
- [20] Juthatip Wisanmongkol et al. “Multipath Mitigation for RSSI-Based Bluetooth Low Energy Localization”. In: *2019 19th International Symposium on Communications and Information Technologies (ISCIT)*. 2019, pp. 47–51. DOI: 10.1109/ISCIT.2019.8905164 (cit. on p. 20).

Online sources

- [21] Android. *Bluetooth*. 2022. URL: <https://source.android.com/docs/core/connect/bluetooth> (cit. on pp. 33, 34).

Check Final Print Size

— Check final print size! —



— Remove this page after printing! —